
Справочник API

Выпуск 2.4.0.214748

янв. 31, 2026

1	C++ API	1
2	C# API	22
3	Python API	42
4	Параметры	88
	Содержание модулей Python	92
	Алфавитный указатель	93

```
class arhiplex_exception : public runtime_error
```

arhiplex-исключение

Исключение, генерируемое всеми классами в случае ошибки

Public Functions

```
inline int get_error_code() const
```

Код ошибки arhiplex. Как правило, всегда -1

```
class Constraint
```

ограничение

Работа с ограничениями у модели, позволяет работать с выражением и границами а также удалять его

Public Functions

```
inline Constraint(const LinearExpression &expr, constraint_sense sense, double rhs)
```

создает ограничение вида (expr (\leq , $=$, \geq) constant)

Параметры

- **expr** – **[in]** выражение, являющееся левой частью ограничения
- **sense** – **[in]** переменная, задающая знак (\leq , $=$, \geq)
- **rhs** – **[in]** правая часть ограничения

```
inline Constraint(const QuadExpression &expr, constraint_sense sense, double rhs)
```

создает ограничение вида (expr (\leq , $=$, \geq) constant)

Параметры

- **expr** – **[in]** выражение, являющееся левой частью ограничения
- **sense** – **[in]** переменная, задающая знак (\leq , $=$, \geq)
- **rhs** – **[in]** правая часть ограничения

inline *LinearExpression* GetLinearExpression() const
Линейное выражение, связанное с ограничением

Результат

Выражение, связанное с ограничением

inline *QuadExpression* GetQuadExpression() const
Квадратичное выражение, связанное с ограничением

Результат

Выражение, связанное с ограничением

inline void SetUpperBound(double upper_bound)
Задаёт верхнюю границу ограничения

Параметры

upper_bound – **[in]** Новая верхняя граница ограничения

inline double GetUpperBound() const
Получает верхнюю границу ограничения

Результат

Верхнюю границу ограничения

inline void SetLowerBound(double lower_bound)
Задаёт нижнюю границу ограничения

Параметры

lower_bound – **[in]** Новая нижняя граница ограничения

inline double GetLowerBound() const
Получает нижнюю границу ограничения

Результат

Нижнюю границу ограничения

inline double GetRange() const
Получает интервал ограничения: upper_bound - lower_bound

Результат

upper_bound - lower_bound

inline void SetRange(double range)
Задаёт интервал ограничения: lower_bound \leq expression \leq lower_bound + range

Параметры

range – **[in]** интервал для ограничения

`inline void Remove()`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`inline const char *GetName() const`

Получает имя ограничения

Результат

имя ограничения

`inline void SetName(const char *szName)`

Задаёт имя ограничения

Параметры

`szName` – **[in]** имя ограничения

`class LinearExpression`

Линейное выражение

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

Public Functions

`inline LinearExpression(double constant = 0.0)`

Создает пустое выражение с возможностью задать константу

Параметры

`constant` – **[in]** константа в выражении

`inline LinearExpression(const Variable &var, double coeff = 1.0)`

Создает выражение на основе переменной и коэффициента

Параметры

- `var` – **[in]** переменная в выражении
- `coeff` – **[in]** коэффициент переменной в выражении

`inline int GetTermsCount() const`

Возвращает кол-во элементов в выражении

Результат

Кол-во элементов в выражении

`inline bool empty() const`

Возвращает истину, если выражение пустое

Результат

истина, если выражение пустое

`inline void SetName(const char *expr_name)`

Задаёт имя выражения

Параметры

`expr_name` – **[in]** имя выражения

`inline const char *GetName() const`

Получает имя выражения

Результат

имя выражения

`inline Variable GetTermVariable(int i) const`

Возвращает переменную по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

Объект переменной

`inline int GetTermVariableIndex(int i) const`

Возвращает индекс переменной по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

индекс переменной

`inline double GetTermCoeff(int i) const`

Возвращает коэффициент переменной по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

Значение коэффициента переменной

`inline double GetConstant() const`

Возвращает константу в выражении

Результат

значение константы

`inline void SetTermCoeff(int i, double value)`

Задаёт коэффициент переменной по индексу выражения

Параметры

- `i` – **[in]** Индекс элемента в выражении
- `value` – **[in]** Значение коэффициента при переменной в элементе

`inline void SetConstant(double constant)`

Задаёт константу в выражении

Параметры

`constant` – **[in]** значение константы

`inline void AddConstant(double constant)`

Добавляет константу к выражению

Параметры

`constant` – **[in]** значение константы

```
inline void AddTerm(const Variable &var, double coeff = 1.0)
```

Добавляет элемент к выражению

Параметры

- **var** – **[in]** переменная
- **coeff** – **[in]** коэффициент к переменной

```
inline void AddExpression(const LinearExpression &expr, double mult = 1.0)
```

Добавляет выражение к выражению

Параметры

- **expr** – **[in]** добавляемое выражение
- **mult** – **[in]** коэффициент к добавляемому выражению

```
inline void RemoveTerm(int idx)
```

Удаляет элемент выражения по индексу

Параметры

idx – **[in]** индекс удаляемого элемента в выражении

```
inline void RemoveVariable(const Variable &var)
```

Удаляет переменную из выражения

Параметры

var – **[in]** удаляемая переменная

```
inline LinearExpression CreateFreeCopy() const
```

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

```
class QuadExpression
```

Квадратичное выражение

Обеспечивает работу с квадратичными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * переменная * коэффициент.

Public Functions

```
inline QuadExpression()
```

Создает пустое выражение

```
inline QuadExpression(const Variable &var1, const Variable &var2, double coeff = 1.0)
```

Создает выражение на основе переменных и коэффициента

Параметры

- **var1** – **[in]** переменная №1 в выражении
- **var2** – **[in]** переменная №2 в выражении
- **coeff** – **[in]** коэффициент в выражении

`inline int GetTermsCount() const`

Возвращает кол-во элементов в выражении

Результат

Кол-во элементов в выражении

`inline bool empty() const`

Возвращает истину, если выражение пустое

Результат

истина, если выражение пустое

`inline void SetName(const char *expr_name)`

Задаёт имя выражения

Параметры

`expr_name` – **[in]** имя выражения

`inline const char *GetName() const`

Получает имя выражения

Результат

имя выражения

`inline Variable GetTermVariable1(int i) const`

Возвращает переменную №1 по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

Объект переменной

`inline Variable GetTermVariable2(int i) const`

Возвращает переменную №2 по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

Объект переменной

`inline double GetTermCoeff(int i) const`

Возвращает коэффициент переменных по индексу элемента в выражении

Параметры

`i` – **[in]** индекс элемента в выражении

Результат

Значение коэффициента

`inline void SetTermCoeff(int i, double value)`

Задаёт коэффициент переменной по индексу выражения

Параметры

- `i` – **[in]** Индекс элемента в выражении
- `value` – **[in]** Значение коэффициента при переменных в элементе


```
inline void AddTerm(const Variable &var1, const Variable &var2, double coeff = 1.0)
```

Добавляет элемент к выражению

Параметры

- `var1` – **[in]** переменная
- `var2` – **[in]** переменная
- `coeff` – **[in]** коэффициент

```
inline void AddExpression(const QuadExpression &expr, double mult = 1.0)
```

Добавляет выражение к выражению

Параметры

- `expr` – **[in]** квадратичное выражение
- `mult` – **[in]** коэффициент к добавляемому выражению

```
inline void AddExpression(const LinearExpression &expr, double mult = 1.0)
```

Добавляет выражение к выражению

Параметры

- `expr` – **[in]** добавляемое выражение
- `mult` – **[in]** коэффициент к добавляемому выражению

```
inline void RemoveTerm(int idx)
```

Удаляет элемент выражения по индексу

Параметры

- `idx` – **[in]** индекс удаляемого элемента в выражении

```
inline QuadExpression CreateFreeCopy() const
```

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

```
class Model
```

Класс для работы с моделью

Предоставляет функции чтения/записи файлов, модификации модели, управления переменными, ограничениями, а также целевой функцией

Public Functions

```
inline Model(const char *name = nullptr)
```

Создает экземпляр модели

Параметры

- `name` – **[in]** Имя модели

```
inline void Read(const char *szFileName)
```

Читает модель из файла, тип модели определяется по расширению

Параметры

`szFileName` – **[in]** путь к файлу модели

`inline void ReadMps(const char *szFileName)`

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

Параметры

`szFileName` – **[in]** путь к файлу модели

`inline void ReadLp(const char *szFileName)`

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

Параметры

`szFileName` – **[in]** путь к файлу модели

`inline int GetIntParam(const char *szParam)`

Получает целочисленный параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

`inline double GetDblParam(const char *szParam)`

Получает параметр с плавающей точкой

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

`inline std::string GetStringParam(const char *szParam)`

Получает строковый параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

`inline bool GetBoolParam(const char *szParam)`

Получает логический параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

`inline void SetIntParam(const char *szParam, int nVal)`

Задаёт целочисленный параметр

Параметры

- `szParam` – **[in]** имя параметра

- **nVal** – **[in]** Новое значение параметра

`inline void SetDblParam(const char *szParam, double dVal)`

Задаёт параметр с плавающей точкой

Параметры

- **szParam** – **[in]** имя параметра
- **dVal** – **[in]** Новое значение параметра

`inline void SetStringParam(const char *szParam, const char *cVal)`

Задаёт строковый параметр

Параметры

- **szParam** – **[in]** имя параметра
- **cVal** – **[in]** Новое значение параметра

`inline void SetBoolParam(const char *szParam, bool bVal)`

Задаёт логический параметр

Параметры

- **szParam** – **[in]** имя параметра
- **bVal** – **[in]** Новое значение параметра

`inline void Write(const char *szFileName, const char *name_mapping_file = "")`

Записывает модель в файл. Тип будет определен по расширению

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`inline void WriteMps(const char *szFileName, const char *name_mapping_file = "")`

Записывает модель в файл в формате MPS

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`inline void WriteLp(const char *szFileName, const char *name_mapping_file = "")`

Записывает модель в файл в формате LP

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

```
inline Variable AddVariable(double lb = 0, double ub = inf, double obj = 0.0,  
                             variable_type var_type = variable_type::continuous, const  
                             char *szName = "")
```

Добавляет переменную в модель с заданными параметрами

Параметры

- **lb** – **[in]** нижняя граница переменной
- **ub** – **[in]** верхняя граница переменной
- **obj** – **[in]** коэффициент к переменной в целевой функции
- **var_type** – **[in]** тип переменной
- **szName** – **[in]** имя переменной

Результат

объект новой переменной

```
inline Constraint AddConstraint(const LinearExpression &expr, constraint_sense sense,  
                                double rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **expr** – **[in]** линейное выражение для ограничения
- **sense** – **[in]** тип ограничения (\leq , \geq , $=$)
- **rhs** – **[in]** константное значение в правой части ограничения
- **szName** – **[in]** имя ограничения. Если передана пустая строка или nullptr, имя ограничения будет сгенерировано

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const QuadExpression &expr, constraint_sense sense,  
                                double rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **expr** – **[in]** квадратичное выражение для ограничения
- **sense** – **[in]** тип ограничения (\leq , \geq , $=$)
- **rhs** – **[in]** константное значение в правой части ограничения
- **szName** – **[in]** имя ограничения. Если передана пустая строка или nullptr, имя ограничения будет сгенерировано

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const LinearExpression &lhs, constraint_sense sense,  
                                const LinearExpression &rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **lhs** – **[in]** выражение для ограничения (левая часть)
- **sense** – **[in]** переменная знака ограничения (\leq , \geq , $=$)
- **rhs** – **[in]** выражение в правой части ограничения
- **szName** – **[in]** имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const LinearExpression &expr, double lb, double ub,  
                               const char *szName)
```

Добавляет интервальное ограничение в модель с заданными параметрами, вида
 $lhs \leq expr \leq rhs$

Параметры

- **expr** – **[in]** выражение для ограничения
- **lb** – **[in]** нижняя граница ограничения
- **ub** – **[in]** верхняя граница ограничения
- **szName** – **[in]** имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const QuadExpression &expr, double lb, double ub,  
                               const char *szName)
```

Добавляет интервальное ограничение в модель с заданными параметрами, вида
 $lhs \leq expr \leq rhs$

Параметры

- **expr** – **[in]** выражение для ограничения
- **lb** – **[in]** нижняя граница ограничения
- **ub** – **[in]** верхняя граница ограничения
- **szName** – **[in]** имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const Constraint &constr, const char *szName)
```

Добавляет уже существующее ограничение в модель

Параметры

- **constr** – **[in]** ограничение
- **szName** – **[in]** имя ограничения

Результат

объект нового ограничения

```
inline Variable GetVariable(int idx) const
```

Получает переменную модели по индексу

Параметры

`idx` – **[in]** индекс переменной

Результат

объект переменной

```
inline int GetVariablesCount() const
```

Получает количество переменных в модели

Результат

количество переменных в модели

```
inline Variable GetVariableByName(const char *name)
```

Получает переменную из модели по имени

Параметры

`name` – **[in]** имя переменной

Результат

объект переменной

```
inline Constraint GetConstraint(int idx) const
```

Получает ограничение из модели по индексу

Параметры

`idx` – **[in]** индекс ограничения

Результат

объект ограничения

```
inline int GetConstraintsCount() const
```

Получает количество ограничений модели

Результат

количество ограничений модели

```
inline Constraint GetConstrByName(const char *szName)
```

Получает ограничение из модели по имени

Параметры

`szName` – **[in]** имя ограничения

Результат

объект ограничения

```
inline void SetObjectiveSense(objective_sense sense)
```

Задаёт тип оптимизации целевой функции - максимизация или минимизация

Параметры

`sense` – **[in]** тип оптимизации

```
inline objective_sense GetObjectiveSense() const
```

Получает тип оптимизации целевой функции

Результат

тип оптимизации

```
inline void SetObjectiveOffset(double constant)
```

Задаёт константу в выражении целевой функции

Параметры

`constant` – **[in]** значение константы в целевой функции

```
inline void SetObjective(const LinearExpression &expr, objective_sense sense =  
                        objective_sense::minimize)
```

Задаёт выражение целевой функции и тип оптимизации

Параметры

- `expr` – **[in]** линейное выражение
- `sense` – **[in]** тип оптимизации

```
inline void SetObjective(const QuadExpression &expr, objective_sense sense =  
                        objective_sense::minimize)
```

Задаёт выражение целевой функции и тип оптимизации

Параметры

- `expr` – **[in]** квадратичное выражение
- `sense` – **[in]** тип оптимизации

```
inline LinearExpression GetObjective() const
```

Получает выражение целевой функции

Результат

Объект выражения

```
inline SolveResult Solve()
```

Стартует оптимизацию модели

Результат

Объект с результатом оптимизации

```
inline SolveResult SolveRemote(const char *name_mapping_file)
```

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения `X_API_KEY`.

Параметры

`name_mapping_file` – **[in]** путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

Результат

Объект с результатом оптимизации

```
inline void SolveRemoteAsync(const char *name_mapping_file)
```

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения `X_API_KEY`.

Параметры

`name_mapping_file` – **[in]** путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

```
inline const char *GetRemoteSolveLog(const char *szCalcUID) const
```

Получает лог удалённого расчёта

Параметры

szCalcUID – **[in]** идентификатор удалённого расчёта

Результат

лог удалённого расчёта

```
inline void Remove(Variable &var)
```

Удаляет переменную из модели

Параметры

var – **[in]** переменная

```
inline void Remove(Constraint &constr)
```

Удаляет ограничение из модели

Параметры

constr – **[in]** ограничение

```
inline void Clear()
```

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

```
inline void SetLogFile(const char *szLogFile)
```

Задать файл для записи лога решения

Параметры

szLogFile – **[in]** файл лога

```
inline void SetLogCallback(ILogCallback *pcb)
```

Задать интерфейс для обратного вызова при записи лога

Параметры

pcb – **[in]** интерфейс для обратного вызова

```
inline void AddMipStartValue(const char *var_name, double var_value)
```

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- var_name – **[in]** имя переменной
- var_value – **[in]** значение переменной

```
inline void AddMipStartValue(const Variable &var, double var_value)
```

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- var – **[in]** переменная
- var_value – **[in]** значение переменной


```
inline void AddMipStartValues(const char *sol_file)
```

Добавить возможные значения переменных в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

`sol_file` – **[in]** путь к файлу с решением

```
inline void ClearMipStartValues()
```

Удалить все стартовые значения для поиска решения

```
inline const char *GetName() const
```

Получает имя модели

Результат

имя модели

```
inline void SetName(const char *szName)
```

Задаёт имя модели

Параметры

`szName` – **[in]** имя модели

```
inline std::string GetCalcUID() const
```

Получить уникальный идентификатор последнего удаленного расчета.

Результат

идентификатор удалённого расчета

```
inline bool IsMip() const
```

Проверяет является ли задача целочисленной.

Результат

true если задача целочисленная, иначе false.

```
inline bool IsNonlinear() const
```

Проверяет является ли задача нелинейной.

Результат

true если задача нелинейная, иначе false.

```
class SolveResult
```

Класс для работы с итогами расчета

Предоставляет доступ к информации о расчете (количество итераций, статус, значение целевой функции и пр.), а также к значениям переменных

Public Functions

```
inline solve_result GetSolveResult() const
```

Получает статус процесса расчетов

Результат

Значение статуса

inline *solution_status* GetSolutionStatus() const

Получает статус модели в итоге расчета

Результат

Значение статуса

inline double GetRelativeGap() const

Получает точность решения в процентах

Результат

точность решения в процентах

inline unsigned int GetIterationsCount() const

Получает количество итераций, проведенных в процессе расчета

Результат

Кол-во итераций

inline std::int64_t GetProcessedNodesCount() const

Получает количество обработанных узлов дерева решений

Результат

Кол-во обработанных узлов

inline double GetObjectiveFunctionValue() const

Получает значение целевой функции

Результат

Значение целевой функции

inline double GetBestBoundValue() const

Получает значение граничной (двойственной) функции

Результат

Значение граничной (двойственной) функции

inline double GetSolveTime() const

Получает общее время решения

Результат

Общее время решения, секунды

inline double GetVariableValue(const *Variable* &var) const

Получает значение переменной в решении

Параметры

var – **[in]** переменная

Результат

Значение переменной в решении

inline double GetVariableValue(const char *var_name) const

Получает значение переменной в решении по имени

Параметры

var_name – **[in]** имя переменной

Результат

Значение переменной в решении

inline double GetDualValue(const *Constraint* &constr) const

Получает значение dual value

Параметры

constr – [in] ограничение

Результат

Значение dual value

inline double GetDualValue(const char *constr_name) const

Получает значение dual value по имени ограничения

Параметры

constr_name – [in] имя ограничения

Результат

Значение dual value

inline double GetReducedCost(const *Variable* &var) const

Получает значение reduced cost

Параметры

var – [in] переменная

Результат

Значение reduced cost

inline double GetReducedCost(const char *var_name) const

Получает значение reduced cost по имени переменной

Параметры

var_name – [in] имя переменной

Результат

Значение reduced cost

inline double GetExpressionValue(const *LinearExpression* &expr) const

Получает значение выражения

Параметры

expr – [in] линейное выражение

Результат

Значение выражения

inline double GetExpressionValue(const *QuadExpression* &expr) const

Получает значение выражения

Параметры

expr – [in] квадратичное выражение

Результат

Значение выражения

inline void WriteSolution(const char *file_name) const

Записывает решение в файл

Параметры

file_name – [in] путь к файлу решения для записи

class **Variable**

Переменная

Работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

Public Functions

inline const char ***GetName**() const

Получает имя переменной в модели

Результат

имя переменной

inline void **SetName**(const char *szName)

Задаёт имя переменной в модели

Параметры

szName – **[in]**

inline arhiplex::*variable_type* **GetType**() const

Получает тип переменной

Результат

тип переменной

inline void **SetType**(arhiplex::*variable_type* type)

Задаёт тип переменной в модели

Параметры

type – **[in]** тип переменной

inline void **SetUpperBound**(double upper_bound)

Задаёт верхнюю границу для переменной

Параметры

upper_bound – **[in]** новая верхняя граница переменной

inline double **GetUpperBound**() const

Получает верхнюю границу переменной

Результат

верхняя граница переменной

inline void **SetLowerBound**(double lower_bound)

Задаёт нижнюю границу для переменной

Параметры

lower_bound – **[in]** новая нижняя граница переменной

inline double **GetLowerBound**() const

Получает нижнюю границу переменной

Результат

нижняя граница переменной

`inline void Remove()`

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

`enum class arhiplex::variable_type`

Тип переменной

Values:

`enumerator continuous`

непрерывная

`enumerator binary`

бинарная

`enumerator integer`

целочисленная

`enumerator semicontinuous`

полу-непрерывная

`enumerator semiinteger`

полу-целая

`enum class arhiplex::objective_sense`

Направление оптимизации целевой функции

Values:

`enumerator minimize`

Минимизация

`enumerator maximize`

Максимизация

`enum class arhiplex::constraint_sense`

Знак ограничения между левой частью (выражением) и правой частью (константой)

Values:

`enumerator equal`

„==“

`enumerator less_equal`

„<=“

enumerator **greater_equal**

„>=“

enum class **arhiplex::solve_result**

Результат процесса расчетов

Values:

enumerator **success**

Процесс решения успешен - найдено некоторое решение или обнаружена достижимость модели

enumerator **fail**

Процесс решения неудачен - не найдено никаких решений и не обнаружена достижимость модели

enumerator **remote_invalid_api_key**

Невалидный ключ для удаленного расчета

enumerator **remote_api_key_not_set**

Переменная окружения X_API_KEY не установлена

enumerator **remote_time_amount_is_over**

Не осталось доступного времени для осуществления удаленного расчета

enumerator **remote_time_per_calc_violated**

Превышен лимит времени для единичного удаленного расчета. Параметр `time_limit` нарушает ограничения лицензии

enumerator **remote_fail**

Удаленный расчет неудачен

enum class **arhiplex::solution_status**

Статус решения по результатам расчета

Values:

enumerator **invalid_solution**

неопределенное/невалидное значение

enumerator **optimal**

решение оптимально (погрешность в рамках заданного значения)

enumerator **feasible**

решение найдено, но не оптимально (погрешность > заданного значения)

`enumerator infeasible`

модель не имеет решения (решение недостижимо)

`enumerator unbounded`

модель неограничена (целевая функция может бесконечно неограниченно увеличиваться/уменьшаться)

`enumerator infeasible_or_unbounded`

модель недостижима или неограничена

```
class arhiplex.ArhiplexException : Exception
```

Исключение, генерируемое всеми классами в случае ошибки

```
class arhiplex.Constraint : IDisposable
```

Работа с ограничениями у модели, позволяет работать с выражением и границами

Public Functions

```
string? GetName ()
```

Получает имя ограничения

```
return
```

имя ограничения

```
void SetName (string name)
```

Задаёт имя ограничения в модели

```
param name
```

имя ограничение

```
void Remove ()
```

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

```
LinearExpression GetLinearExpression ()
```

Линейное выражение, связанное с ограничением

```
return
```

Линейное выражение

```
QuadExpression GetQuadExpression ()
```

Квадратичное выражение, связанное с ограничением

return

Квадратичное выражение

void **SetUpperBound** (double *upper_bound*)

Задаёт верхнюю границу для ограничения

param upper_bound

новая верхняя граница ограничения

double **GetUpperBound** ()

Получает верхнюю границу ограничения

return

верхняя граница ограничения

void **SetLowerBound** (double *lower_bound*)

Задаёт нижнюю границу для ограничения

param lower_bound

новая нижняя граница ограничения

double **GetLowerBound** ()

Получает нижнюю границу ограничения

return

нижняя граница ограничения

double **GetRange** ()

Получает интервал ограничения: *upper_bound* - *lower_bound*

return

upper_bound - *lower_bound*

void **SetRange** (double *range*)

Задаёт интервал ограничения: *lower_bound* <= *expression* <= *lower_bound* + *range*

param range

интервал для ограничения

class **arhiplex.LinearExpression** : **IDisposable**

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

Public Functions

LinearExpression (double *offset*)

Создает пустое выражение с возможностью задать константу

param offset

константа в выражении

LinearExpression (*Variable* *var*, double *coeff* = 1.0)

Создает выражение на основе переменной и коэффициента

param var

переменная в выражении

param coeff

коэффициент переменной в выражении

`int GetTermsCount ()`

Возвращает кол-во элементов в выражении

return

кол-во элементов в выражении

Variable `GetTermVariable (int term_idx)`

Возвращает переменную по индексу элемента в выражении

param term_idx

индекс элемента в выражении

return

Объект переменной

`double GetTermCoeff (int term_idx)`

Возвращает коэффициент переменной по индексу элемента в выражении

param term_idx

индекс элемента в выражении

return

Значение коэффициента переменной

`void SetTermCoeff (int term_idx, double value)`

Задаёт коэффициент переменной по индексу выражения

param term_idx

индекс элемента в выражении

param value

Значение коэффициента при переменной в элементе

`double GetConstant ()`

Возвращает константу в выражении

return

значение константы

`void SetConstant (double constant)`

Задаёт константу в выражении

param constant

значение константы

`void AddConstant (double constant)`

Добавляет константу к выражению

param constant

значение константы

void AddTerm (*Variable* var, double coeff)

Добавляет элемент к выражению

param var

переменная

param coeff

коэффициент к переменной

void AddExpression (*LinearExpression* expr, double mult = 1.0)

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

void RemoveTerm (int term_idx)

Удаляет элемент выражения по индексу

param term_idx

индекс удаляемого элемента в выражении

void RemoveVariable (*Variable* var)

Удаляет переменную из выражения

param var

удаляемая переменная

QuadExpression GetQuadPart ()

Возвращает квадратичную часть выражения

return

квадратичная часть выражения

LinearExpression CreateFreeCopy ()

Создает копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

class arhiplex.QuadExpression : IDisposable

Обеспечивает работу с квадратичными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * переменная * коэффициент.

Public Functions

QuadExpression ()

Создает пустое выражение

QuadExpression (Variable var1, Variable var2, double coeff = 1.0)

Создает выражение на основе переменных и коэффициента

param var1

переменная №1 в выражении

param var2

переменная №2 в выражении

param coeff

коэффициент в выражении

int GetTermsCount ()

Возвращает кол-во элементов в выражении

return

Кол-во элементов в выражении

Variable GetTermVariable1 (int term_idx)

Возвращает переменную №1 по индексу элемента в выражении

param term_idx

индекс элемента в выражении

return

объект переменной

Variable GetTermVariable2 (int term_idx)

Возвращает переменную №2 по индексу элемента в выражении

param term_idx

индекс элемента в выражении

return

объект переменной

double GetTermCoeff (int term_idx)

Возвращает коэффициент переменных по индексу элемента в выражении

param term_idx

индекс элемента в выражении

return

Значение коэффициента

void SetTermCoeff (int term_idx, double value)

Задаёт коэффициент переменной по индексу выражения

param term_idx

индекс элемента в выражении

param value

Значение коэффициента при переменных в элементе

void AddTerm (*Variable* var1, *Variable* var2, double coeff)

Добавляет элемент к выражению

param var1

переменная

param var2

переменная

param coeff

коэффициент

void AddExpression (*QuadExpression* quad_expr, double mult = 1.0)

Добавляет выражение к выражению

param quad_expr

квадратичное выражение

param coeff

коэффициент к добавляемому выражению

void RemoveTerm (int term_idx)

Удаляет элемент выражения по индексу

param term_idx

индекс удаляемого элемента в выражении

LinearExpression GetLinearPart ()

Возвращает линейную часть выражения

return

линейная часть

QuadExpression CreateFreeCopy ()

Создает копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

class arhiplex.Model : IDisposable

Предоставляет функции чтения/записи файлов, модификации модели, управления переменными, ограничениями, а также целевой функцией

Public Functions

Model (string name = "")

Создает экземпляр модели

param name

Имя модели

`void Read (string file_name)`

Читает модель из файла, тип модели определяется по расширению

param *file_name*
путь к файлу модели

`void ReadMps (string file_name)`

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

param *file_name*
путь к файлу модели

`void ReadLp (string file_name)`

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

param *file_name*
путь к файлу модели

`void Write (string file_name, string name_mapping_file = "")`

Записывает модель в файл. Тип будет определен по расширению

param *file_name*
путь к записываемому файлу

param *name_mapping_file*
путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`void WriteMps (string file_name, string name_mapping_file = "")`

Записывает модель в файл в формате MPS

param *file_name*
путь к записываемому файлу

param *name_mapping_file*
путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`void WriteLp (string file_name, string name_mapping_file = "")`

Записывает модель в файл в формате LP

param *file_name*
путь к записываемому файлу

param *name_mapping_file*
путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`int GetIntParam (string param_name)`

Получает целочисленный параметр

param *param_name*
имя параметра

return

Значение параметра

double **GetDblParam** (string *param_name*)

Получает параметр с плавающей точкой

param param_name

имя параметра

return

Значение параметра

string **GetStringParam** (string *param_name*)

Получает строковый параметр

param param_name

имя параметра

return

Значение параметра

bool **GetBoolParam** (string *param_name*)

Получает логический параметр

param param_name

имя параметра

return

Значение параметра

void **SetIntParam** (string *param_name*, int *nVal*)

Задаёт целочисленный параметр

param param_name

имя параметра

param nVal

Новое значение параметра

void **SetDblParam** (string *param_name*, double *dVal*)

Задаёт параметр с плавающей точкой

param param_name

имя параметра

param dVal

Новое значение параметра

void **SetStringParam** (string *param_name*, string *cVal*)

Задаёт строковый параметр

param param_name

имя параметра

param cVal

Новое значение параметра

void SetBoolParam (string *param_name*, bool *bVal*)

Задаёт логический параметр

param *param_name*

имя параметра

param *bVal*

Новое значение параметра

string? GetName ()

Получает имя модели

return

Имя модели

void SetName (string *name*)

Задаёт имя модели

param *name*

Имя модели

Variable AddVariable (double *lb*, double *ub*, double *obj*, *Variable_type* *var_type*,
string *name*)

Добавляет переменную в модель с заданными параметрами

param *lb*

нижняя граница переменной

param *ub*

верхняя граница переменной

param *obj*

коэффициент к переменной в целевой функции

param *var_type*

тип переменной

param *name*

имя переменной

Constraint AddConstraint (*LinearExpression* *lin_expr*, *Constraint_sense* *sense*,
double *rhs*, string *name*)

Добавляет ограничение в модель с заданными параметрами

param *lin_expr*

линейное выражение для ограничения

param *sense*

тип ограничения (<=, >= , ==)

param *rhs*

константное значение в правой части ограничения

param *name*

имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

return

объект нового ограничения

Constraint AddConstraint (*QuadExpression* quad_expr, *Constraint_sense* sense, double rhs, string name)

Добавляет ограничение в модель с заданными параметрами

param quad_expr

квадратичное выражение для ограничения

param sense

тип ограничения (<=, >=, ==)

param rhs

константное значение в правой части ограничения

param name

имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

return

объект нового ограничения

Constraint AddConstraint (*LinearExpression* lin_expr, double lb, double ub, string name)

Добавляет ограничение в модель с заданными параметрами

param lin_expr

линейное выражение для ограничения

param lb

нижняя граница ограничения

param ub

верхняя граница ограничения

param name

имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

return

объект нового ограничения

Constraint AddConstraint (*QuadExpression* quad_expr, double lb, double ub, string name)

Добавляет ограничение в модель с заданными параметрами

param quad_expr

квадратичное выражение для ограничения

param lb

нижняя граница ограничения

param ub

верхняя граница ограничения

param name

имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

return

объект нового ограничения

Constraint AddConstraint (*Constraint* constr, string name)

Добавляет уже существующее ограничение в модель

param constr

ограничение

param name

имя ограничения

return

объект нового ограничения

int GetVariablesCount ()

Получает количество переменных в модели

return

количество переменных в модели

Variable GetVariable (int var_idx)

Получает переменную модели по индексу

param var_idx

индекс переменной

return

объект переменной

Variable GetVariable (string name)

Получает переменную модели по имени

param name

имя переменной

return

объект переменной

int GetConstraintsCount ()

Получает количество ограничений модели

return

количество ограничений модели

Constraint GetConstraint (int constr_idx)

Получает ограничение из модели по индексу

param constr_idx

индекс ограничения

return

объект ограничения

Constraint GetConstraint (string name)

Получает ограничение из модели по имени

param name
имя ограничения

return
объект ограничения

void SetObjectiveSense (*Objective_sense* sense)

Задаёт тип оптимизации целевой функции - максимизация или минимизация

param sense
тип оптимизации

Objective_sense GetObjectiveSense ()

Получает тип оптимизации целевой функции

return
тип оптимизации целевой функции

void SetObjectiveOffset (double offset)

Задаёт константу в выражении целевой функции

param offset
значение константы в целевой функции

void SetObjective (*LinearExpression* expr, *Objective_sense* sense =
Objective_sense.minimize)

Задаёт выражение целевой функции и тип оптимизации

param expr
линейное выражение

param sense
тип оптимизации

void SetObjective (*QuadExpression* quad_expr, *Objective_sense* sense =
Objective_sense.minimize)

Задаёт выражение целевой функции и тип оптимизации

param expr
квадратичное выражение

param sense
тип оптимизации

void ClearObjective ()

Удаляет целевую функцию из модели

LinearExpression GetObjective ()

Получает выражение целевой функции

return
Объект выражения

QuadExpression GetQuadObjective ()

Получает выражение целевой функции

return

Объект выражения

SolveResult Solve ()

Стартует оптимизацию модели

return

Объект с результатом оптимизации

SolveResult SolveRemote (string *name_mapping_file*)

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

param name_mapping_file

путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

return

Объект с результатом оптимизации

void SolveRemoteAsync (string *name_mapping_file*)

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения X_API_KEY.

param name_mapping_file

путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

string? GetRemoteSolveLog (string *calc_uid*)

Получает лог удалённого расчета

param calc_uid

идентификатор удалённого расчета

return

лог удалённого расчета

void Remove (*Variable* *var*)

Удаляет переменную из модели

param var

переменная

void Remove (*Constraint* *constr*)

Удаляет ограничение из модели

param constr

ограничение

void Clear ()

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

void SetLogFile (string *log_file*)

Задать файл для записи лога решения

param log_file

файл лога

void AddMipStartValue (string *var_name*, double *var_value*)

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

param var_name

имя переменной

param var_value

значение переменной

void AddMipStartValue (*Variable* *var*, double *var_value*)

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

param var

переменная

param var_value

значение переменной

void AddMipStartValues (string *sol_file*)

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

param sol_file

путь к файлу с решением

void ClearMipStartValues ()

Удалить все стартовые значения для поиска решения

string? GetCalcUID ()

Получить уникальный идентификатор последнего удаленного расчета.

return

идентификатор удаленного расчета.

bool IsMip ()

Проверяет является ли задача целочисленной.

return

true если задача целочисленная, иначе false.

bool IsNonlinear ()

Проверяет является ли задача нелинейной.

return

true если задача нелинейная, иначе false.

class arhiplex.SolveResult : IDisposable

Предоставляет доступ к информации о расчете (количество итераций, статус, значение целевой функции и пр.), а также к значениям переменных

Public Functions

Solve_result GetSolveResult ()

Получает статус процесса расчетов

return
Значение статуса

Solution_status GetSolutionStatus ()

Получает статус модели в итоге расчета

return
Значение статуса

double GetRelativeGap ()

Получает точность решения в процентах

return
точность решения в процентах

uint GetIterationsCount ()

Получает количество итераций, проведенных в процессе расчета

return
Кол-во итераций

Int64 GetProcessedNodesCount ()

Получает количество обработанных узлов дерева решений

return
Кол-во обработанных узлов

double GetObjectiveFunctionValue ()

Получает значение целевой функции

return
Значение целевой функции

double GetBestBoundValue ()

Получает значение граничной (двойственной) функции

return
Значение граничной (двойственной) функции

double GetSolveTime ()

Получает общее время решения

return
Общее время решения, секунды

double GetVariableValue (*Variable* var)

Получает значение переменной в решении

param var
объект переменной

return
Значение переменной в решении

double GetVariableValue (string *var_name*)

Получает значение переменной в решении по имени

param var_name
имя переменной

return
Значение переменной в решении

double GetDualValue (*Constraint* *constr*)

Получает значение dual value

param constr
ограничение

return
Значение dual value

double GetDualValue (string *constr_name*)

Получает значение dual value по имени ограничения

param constr_name
имя ограничения

return
Значение dual value

double GetReducedCost (*Variable* *var*)

Получает значение reduced cost

param var
переменная

return
Значение reduced cost

double GetReducedCost (string *var_name*)

Получает значение reduced cost

param var
имя переменной

return
Значение reduced cost

double GetExpressionValue (*LinearExpression* *expression*)

Получает значение выражения

param expression
линейное выражение

return
Значение выражения

double GetExpressionValue (*QuadExpression* expression)

Получает значение выражения

param expression

квадратичное выражение

return

Значение выражения

void WriteSolution (string file_name)

Записывает решение в файл

param file_name

путь к файлу решения для записи

class arhiplex.Variable : IDisposable

Класс для работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

Public Functions

void SetUpperBound (double upper_bound)

Задаёт верхнюю границу для переменной

param upper_bound

новая верхняя граница переменной

double GetUpperBound ()

Получает верхнюю границу переменной

return

верхняя граница переменной

void SetLowerBound (double lower_bound)

Задаёт нижнюю границу для переменной

param lower_bound

новая нижняя граница переменной

double GetLowerBound ()

Получает нижнюю границу переменной

return

нижняя граница переменной

void Remove ()

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

Variable_type GetVarType ()

Получает тип переменной

return

тип переменной

void **SetType** (*Variable_type* type)

Задаёт тип переменной в модели

param type

тип переменной

string? **GetName** ()

Получает имя переменной

return

имя переменной

void **SetName** (string name)

Задаёт имя переменной в модели

param name

имя переменной

enum arhiplex.Variable_type

Тип переменной

Values:

continuous

непрерывная

binary

бинарная

integer

целочисленная

semicontinuous

полу-непрерывная

semiinteger

полу-целая

enum arhiplex.Objective_sense

Направление оптимизации целевой функции

Values:

minimize

Минимизация

maximize

Максимизация

enum arhiplex.Constraint_sense

Знак ограничения между левой частью (выражением) и правой частью (константой)

Values:

equal

„==“

less_equal

„<=“

greater_equal

„>=“

enum arhiplex.Solve_result

Результат процесса расчетов

Values:

success

Процесс решения успешен - найдено некоторое решение или обнаружена недо-
стижимость модели

fail

Процесс решения неудачен - не найдено никаких решений и не обнаружена недо-
стижимость модели

remote_invalid_api_key

Невалидный ключ для удаленного расчета

remote_api_key_not_set

Переменная окружения X_API_KEY не установлена

remote_time_amount_is_over

Не осталось доступного времени для осуществления удаленного расчета

remote_time_per_calc_violated

Превышен лимит времени для единичного удаленного расчета. Параметр
time_limit нарушает ограничения лицензии

remote_fail

Удаленный расчет неудачен

enum arhiplex.Solution_status

Статус решения по результатам расчета

Values:

invalid_solution

неопределенное/невалидное значение

optimal

решение оптимально (погрешность в рамках заданного значения)

feasible

решение найдено, но не оптимально (погрешность $>$ заданного значения)

infeasible

модель не имеет решения (решение недостижимо)

unbounded

модель неограничена (целевая функция может бесконечно неограниченно увеличиваться/уменьшаться)

infeasible_or_unbounded

модель недостижима или неограничена

exception arhiplexpy.ArhiplexException

class arhiplexpy.Constraint

Работа с линейными ограничениями у модели,

позволяет работать с выражением и границами, а также удалять его

`get_linear_expression(self: arhiplexpy.Constraint) → arhiplex::LinearExpression`

Линейная часть выражения, связанного с ограничением

return

линейная часть выражения, связанного с ограничением

`get_lower_bound(self: arhiplexpy.Constraint) → float`

Получает нижнюю границу ограничения

return

нижнюю границу ограничения

`get_name(self: arhiplexpy.Constraint) → str`

Получает имя ограничения

return

имя ограничения

`get_quad_expression(self: arhiplexpy.Constraint) → arhiplex::QuadExpression`

Квадратичная часть выражения, связанного с ограничением

return

квадратичная часть выражения, связанного с ограничением

`get_range(self: arhiplexpy.Constraint) → float`

Получает интервал ограничения: `upper_bound - lower_bound`

return

`upper_bound - lower_bound`

`get_upper_bound(self: arhiplexpy.Constraint) → float`

Получает верхнюю границу ограничения

return

верхнюю границу ограничения

`remove(self: arhiplexpy.Constraint) → None`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`set_lower_bound(self: arhiplexpy.Constraint, lower_bound: float) → None`

Задаёт нижнюю границу ограничения

param lower_bound

новая нижняя граница ограничения

`set_name(self: arhiplexpy.Constraint, constr_name: str) → None`

Задаёт имя ограничения

param constr_name

имя ограничения

`set_range(self: arhiplexpy.Constraint, range: float) → None`

Задаёт интервал ограничения: $\text{lower_bound} \leq \text{expression} \leq \text{lower_bound} + \text{range}$

param range

интервал для ограничения

`set_upper_bound(self: arhiplexpy.Constraint, upper_bound: float) → None`

Задаёт верхнюю границу ограничения

param upper_bound

новая верхняя граница ограничения

`class arhiplexpy.GeneralConstraint`

Работа с нелинейными ограничениями у модели

`get_expression(self: arhiplexpy.GeneralConstraint) → arhiplex::GeneralExpression`

Выражение, связанное с ограничением

return

выражение, связанное с ограничением

`get_name(self: arhiplexpy.GeneralConstraint) → str`

Получает имя ограничения

return

имя ограничения

`get_variable(self: arhiplexpy.GeneralConstraint) → arhiplex::Variable`

Переменная, связанная с ограничением

return

переменная, связанная с ограничением

`remove(self: arhiplexpy.GeneralConstraint) → None`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`set_name(self: arhiplexpy.GeneralConstraint, constr_name: str) → None`

Задаёт имя ограничения

param constr_name

имя ограничения

`class arhiplexpy.GeneralExpression`

Выражение общего вида

`add_expr(self: arhiplexpy.GeneralExpression, expr: arhiplexpy.GeneralExpression) →`

`None`

Добавляет выражение, для типов plus и multiply

:param expr : добавляемое выражение

`copy(self: arhiplexpy.GeneralExpression) → arhiplexpy.GeneralExpression`

Создает копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

`get_arity(self: arhiplexpy.GeneralExpression) → int`

Возвращает арность выражения

return

арность выражения

`get_constant(self: arhiplexpy.GeneralExpression) → float`

Возвращает значение константы для выражения с типом constant

return

значение константы

`get_expr(self: arhiplexpy.GeneralExpression, idx: int) → arhiplexpy.GeneralExpression`

Возвращает по индексу содержащееся выражение

:param idx : индекс получаемого выражения :return: выражение

`get_expr_count(self: arhiplexpy.GeneralExpression) → int`

Возвращает количество элементов в выражении

return

количество элементов в выражении

`get_type(self: arhiplexpy.GeneralExpression) → arhiplexpy.gen_expr_type`

Возвращает тип выражения

return

тип выражения

`get_variable(self: arhiplexpy.GeneralExpression) → arhiplexpy.Variable`

Возвращает переменную для выражения с типом variable

return

объект переменной

`set_expr(self: arhiplexpy.GeneralExpression, idx: int, expr: arhiplexpy.GeneralExpression) → None`

Задаёт выражение по индексу

:param idx : индекс выражения :param expr : выражение

`class arhiplexpy.LinearExpression`

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

`add_constant(self: arhiplexpy.LinearExpression, value: float) → None`

Добавляет константу к выражению

param

value значение константы

`add_expression(self: arhiplexpy.LinearExpression, expr: arhiplexpy.LinearExpression, mult: float) → None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

`add_term(self: arhiplexpy.LinearExpression, var: arhiplexpy.Variable, coeff: float = 1.0) → None`

Добавляет элемент к выражению

param var

переменная

param coeff

коэффициент к переменной

`copy(self: arhiplexpy.LinearExpression) → arhiplexpy.LinearExpression`

Создаёт копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

`empty(self: arhiplexpy.LinearExpression) → bool`

Возвращает истину, если выражение пустое

return

истина, если выражение пустое

`get_constant(self: arhiplexpy.LinearExpression) → float`

Возвращает константу в выражении

return

значение константы

`get_name(self: arhiplexpy.LinearExpression) → str`

Получает имя выражения

return

имя выражения

`get_quad_part(self: arhiplexpy.LinearExpression) → arhiplex::QuadExpression`

Возвращает квадратичную часть выражения

return

квадратичная часть выражения

`get_term_coeff(self: arhiplexpy.LinearExpression, ind: int) → float`

Возвращает коэффициент переменной по индексу элемента в выражении

param ind

индекс элемента в выражении

return

значение коэффициента переменной

`get_term_variable(self: arhiplexpy.LinearExpression, ind: int) → arhiplexpy.Variable`

Возвращает переменную по индексу элемента в выражении

param ind

индекс элемента в выражении

return

объект переменной

`get_term_variable_idx(self: arhiplexpy.LinearExpression, ind: int) → int`

Возвращает порядковый индекс переменной в модели по индексу элемента в выражении

param ind

индекс элемента в выражении

return

индекс переменной

`get_terms_count(self: arhiplexpy.LinearExpression) → int`

Возвращает количество элементов в выражении

return

количество элементов в выражении

`remove_term(self: arhiplexpy.LinearExpression, idx: int) → None`

Удаляет элемент выражения по индексу

param idx

индекс удаляемого элемента в выражении

`remove_variable(self: arhiplexpy.LinearExpression, var: arhiplexpy.Variable) → None`

Удаляет переменную из выражения

param var

удаляемая переменная

`set_constant(self: arhiplexpy.LinearExpression, value: float) → None`

Задаёт константу в выражении

param value

значение константы

`set_name(self: arhiplexpy.LinearExpression, expr_name: str) → None`

Задаёт имя выражения

param expr_name

имя выражения

`set_term_coeff(self: arhiplexpy.LinearExpression, ind: int, value: float) → None`

Задаёт коэффициент переменной по индексу выражения

param ind

индекс элемента в выражении

param value

значение коэффициента при переменной в элементе

`class arhiplexpy.MathFunc`

Математические функции для нелинейных функций

`cos(*args, **kwargs)`

Overloaded function.

1. `cos(self: float) -> arhiplexpy.GeneralExpression`Получает выражение с функцией `cos`**Результат**выражение с функцией `cos`2. `cos(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`Получает выражение с функцией `cos`**Результат**выражение с функцией `cos`3. `cos(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`Получает выражение с функцией `cos`**Результат**выражение с функцией `cos`4. `cos(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`Получает выражение с функцией `cos`**Результат**выражение с функцией `cos`5. `cos(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`Получает выражение с функцией `cos`

Результат

выражение с функциейcos

`exp(*args, **kwargs)`

Overloaded function.

1. `exp(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейexp

Результат

выражение с функциейexp

2. `exp(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейexp

Результат

выражение с функциейexp

3. `exp(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейexp

Результат

выражение с функциейexp

4. `exp(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейexp

Результат

выражение с функциейexp

5. `exp(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейexp

Результат

выражение с функциейexp

`log(*args, **kwargs)`

Overloaded function.

1. `log(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog

Результат

выражение с функциейlog

2. `log(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog

Результат

выражение с функциейlog

3. `log(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log`

Результат

выражение с функцией `log`

4. `log(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log`

Результат

выражение с функцией `log`

5. `log(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log`

Результат

выражение с функцией `log`

`log10(*args, **kwargs)`

Overloaded function.

1. `log10(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log10`

Результат

выражение с функцией `log10`

2. `log10(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log10`

Результат

выражение с функцией `log10`

3. `log10(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log10`

Результат

выражение с функцией `log10`

4. `log10(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log10`

Результат

выражение с функцией `log10`

5. `log10(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `log10`

Результат

выражение с функциейlog10

`log2(*args, **kwargs)`

Overloaded function.

1. `log2(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog2

Результат

выражение с функциейlog2

2. `log2(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog2

Результат

выражение с функциейlog2

3. `log2(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog2

Результат

выражение с функциейlog2

4. `log2(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog2

Результат

выражение с функциейlog2

5. `log2(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlog2

Результат

выражение с функциейlog2

`logistic(*args, **kwargs)`

Overloaded function.

1. `logistic(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlogistic

Результат

выражение с функциейlogistic

2. `logistic(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейlogistic

Результат

выражение с функциейlogistic

3. `logistic(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `logistic`

Результат

выражение с функцией `logistic`

4. `logistic(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `logistic`

Результат

выражение с функцией `logistic`

5. `logistic(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `logistic`

Результат

выражение с функцией `logistic`

`pow(*args, **kwargs)`

Overloaded function.

1. `pow(self: float, arg0: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `pow`

Результат

выражение с функцией `pow`

2. `pow(self: float, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `pow`

Результат

выражение с функцией `pow`

3. `pow(self: float, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `pow`

Результат

выражение с функцией `pow`

4. `pow(self: float, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `pow`

Результат

выражение с функцией `pow`

5. `pow(self: float, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейrow

Результат

выражение с функциейrow

6. pow(self: arhiplexpy.Variable, arg0: float) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

7. pow(self: arhiplexpy.Variable, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

8. pow(self: arhiplexpy.Variable, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

9. pow(self: arhiplexpy.Variable, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

10. pow(self: arhiplexpy.Variable, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

11. pow(self: arhiplexpy.LinearExpression, arg0: float) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

12. pow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

13. pow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

14. pow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

15. pow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

16. pow(self: arhiplexpy.QuadExpression, arg0: float) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

17. pow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

18. pow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

19. pow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

20. pow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

21. pow(self: arhiplexpy.GeneralExpression, arg0: float) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

22. pow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

23. pow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

24. pow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

25. pow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейrow

Результат

выражение с функциейrow

`signpow(*args, **kwargs)`

Overloaded function.

1. `signpow(self: float, arg0: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

2. `signpow(self: float, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

3. `signpow(self: float, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

4. `signpow(self: float, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

5. `signpow(self: float, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

6. `signpow(self: arhiplexpy.Variable, arg0: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

7. `signpow(self: arhiplexpy.Variable, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

8. `signpow(self: arhiplexpy.Variable, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

9. `signpow(self: arhiplexpy.Variable, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

10. `signpow(self: arhiplexpy.Variable, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

11. `signpow(self: arhiplexpy.LinearExpression, arg0: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

12. `signpow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

13. `signpow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

14. `signpow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

15. `signpow(self: arhiplexpy.LinearExpression, arg0: arhiplexpy.GeneralExpression)`
-> `arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

16. `signpow(self: arhiplexpy.QuadExpression, arg0: float)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

17. `signpow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.Variable)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

18. `signpow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.LinearExpression)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

19. `signpow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.QuadExpression)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

20. `signpow(self: arhiplexpy.QuadExpression, arg0: arhiplexpy.GeneralExpression)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

21. `signpow(self: arhiplexpy.GeneralExpression, arg0: float)` ->
`arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

22. `signpow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

23. `signpow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

24. `signpow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

25. `signpow(self: arhiplexpy.GeneralExpression, arg0: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `signpow`

Результат

выражение с функцией `signpow`

`sin(*args, **kwargs)`

Overloaded function.

1. `sin(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sin`

Результат

выражение с функцией `sin`

2. `sin(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sin`

Результат

выражение с функцией `sin`

3. `sin(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sin`

Результат

выражение с функцией `sin`

4. `sin(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sin`

Результат

выражение с функцией `sin`

5. `sin(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sin`

Результат

выражение с функцией `sin`

`sqr(*args, **kwargs)`

Overloaded function.

1. `sqr(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sqr`

Результат

выражение с функцией `sqr`

2. `sqr(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sqr`

Результат

выражение с функцией `sqr`

3. `sqr(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sqr`

Результат

выражение с функцией `sqr`

4. `sqr(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sqr`

Результат

выражение с функцией `sqr`

5. `sqr(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `sqr`

Результат

выражение с функцией `sqr`

`sqrt(*args, **kwargs)`

Overloaded function.

1. `sqrt(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функциейsqrt

Результат

выражение с функциейsqrt

2. sqrt(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейsqrt

Результат

выражение с функциейsqrt

3. sqrt(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейsqrt

Результат

выражение с функциейsqrt

4. sqrt(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейsqrt

Результат

выражение с функциейsqrt

5. sqrt(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейsqrt

Результат

выражение с функциейsqrt

`tan(*args, **kwargs)`

Overloaded function.

1. tan(self: float) -> arhiplexpy.GeneralExpression

Получает выражение с функциейtan

Результат

выражение с функциейtan

2. tan(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression

Получает выражение с функциейtan

Результат

выражение с функциейtan

3. tan(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression

Получает выражение с функциейtan

Результат

выражение с функциейtan

4. `tan(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tan`

Результат

выражение с функцией `tan`

5. `tan(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tan`

Результат

выражение с функцией `tan`

`tanh(*args, **kwargs)`

Overloaded function.

1. `tanh(self: float) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tanh`

Результат

выражение с функцией `tanh`

2. `tanh(self: arhiplexpy.Variable) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tanh`

Результат

выражение с функцией `tanh`

3. `tanh(self: arhiplexpy.LinearExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tanh`

Результат

выражение с функцией `tanh`

4. `tanh(self: arhiplexpy.QuadExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tanh`

Результат

выражение с функцией `tanh`

5. `tanh(self: arhiplexpy.GeneralExpression) -> arhiplexpy.GeneralExpression`

Получает выражение с функцией `tanh`

Результат

выражение с функцией `tanh`

`class arhiplexpy.MatrixModel`

Создает экземпляр матричной модели

`add_constraint(self: arhiplexpy.MatrixModel, P: object = None, q: object = None, sense: arhiplexpy.constraint_sense, r: float) → None`

Добавляет ограничение в модель с заданными параметрами

param P

матрица квадратичных коэффициентов

param q

вектор коэффициентов переменных

param sense

тип ограничения (<=, >= , ==)

param r

константное значение в правой части ограничения

`add_constraints(self: arhiplexpy.MatrixModel, A: object, senses: object, b: object) → None`

Добавляет ограничения в модель с заданными параметрами

param A

матрица ограничений в формате csr (sparse row matrix)

param senses

массив типов ограничений (constraint_sense)

param b

массив константных значений в правых частях ограничений

`add_variables(self: arhiplexpy.MatrixModel, lower_bounds: object = 0.0, upper_bounds: object = inf, obj_values: object = 0.0, var_types: object = <variable_type.continuous: 0>, count: int) → None`

Добавляет переменные в модель с заданными параметрами

param lower_bounds

нижние границы переменных

param upper_bounds

верхние границы переменных

param obj_values

коэффициенты к переменным в целевой функции

param var_types

типы переменных

param count

количество переменных

`clear(self: arhiplexpy.MatrixModel) → None`

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

`get_bool_param(self: arhiplexpy.MatrixModel, param_name: str) → bool`

Получает логический параметр

param param_name

имя параметра

return

значение параметра

`get_dbl_param(self: arhiplexpy.MatrixModel, param_name: str) → float`

Получает параметр с плавающей точкой

param param_name

имя параметра

return

значение параметра

`get_int_param(self: arhiplexpy.MatrixModel, param_name: str) → int`

Получает целочисленный параметр

param param_name

имя параметра

return

значение параметра

`get_string_param(self: arhiplexpy.MatrixModel, param_name: str) → str`

Получает строковый параметр

param param_name

имя параметра

return

значение параметра

`get_variables_count(self: arhiplexpy.MatrixModel) → int`

Получает количество переменных в модели

return

количество переменных в модели

`set_bool_param(self: arhiplexpy.MatrixModel, param_name: str, param_value: bool) → None`

Задаёт логический параметр

param param_name

имя параметра

param param_value

значение параметра

`set_dbl_param(self: arhiplexpy.MatrixModel, param_name: str, param_value: float) → None`

Задаёт параметр с плавающей точкой

param param_name

имя параметра

param param_value

значение параметра

`set_int_param(self: arhiplexpy.MatrixModel, param_name: str, param_value: int) → None`

Задаёт целочисленный параметр

param param_name

имя параметра

param param_value

значение параметра

set_log_file(*self*: [arhiplexpy.MatrixModel](#), *log_file*: *str*) → None

Задать файл для записи лога решения

:param log_file файл лога

set_objective(*self*: [arhiplexpy.MatrixModel](#), *P*: *object* = None, *q*: *object* = None, *sense*: [arhiplexpy.objective_sense](#) = <*objective_sense.minimize*: 0>) → None

Задаёт выражение целевой функции и тип оптимизации

param P

квадратичная часть выражения

param q

вектор коэффициентов к переменным

param sense

тип оптимизации

set_objective_offset(*self*: [arhiplexpy.MatrixModel](#), *offset*: *float*) → None

Задаёт константу в выражении целевой функции

param offset

значение константы

set_string_param(*self*: [arhiplexpy.MatrixModel](#), *param_name*: *str*, *param_value*: *str*) → None

Задаёт строковый параметр

param param_name

имя параметра

param param_value

значение параметра

solve(*self*: [arhiplexpy.MatrixModel](#)) → [arhiplex::MatrixSolveResult](#)

Стартует оптимизацию модели

return

объект с результатом оптимизации

solve_remote(*self*: [arhiplexpy.MatrixModel](#)) → [arhiplex::MatrixSolveResult](#)

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

return

объект с результатом оптимизации

write(*self*: [arhiplexpy.MatrixModel](#), *file_name*: *str*) → None

Записывает модель в файл. Тип будет определен по расширению

param file_name

путь к записываемому файлу

`write_lp(self: arhiplexpy.MatrixModel, file_name: str) → None`

Записывает модель в файл в формате LP

param file_name

путь к записываемому файлу

`write_mps(self: arhiplexpy.MatrixModel, file_name: str) → None`

Записывает модель в файл в формате MPS

param file_name

путь к записываемому файлу

`class arhiplexpy.MatrixSolveResult`

Результат расчета матричной модели

`get_best_bound(self: arhiplexpy.MatrixSolveResult) → float`

Получает значение граничной (двойственной) функции

return

значение граничной (двойственной) функции

`get_dual_values_vector(self: arhiplexpy.MatrixSolveResult) →`

`numpy.ndarray[numpy.float64]`

Получает dual values вектор

return

вектор dual values

`get_iterations_count(self: arhiplexpy.MatrixSolveResult) → int`

Получает количество итераций, проведенных в процессе расчета

return

количество итераций

`get_nodes_count(self: arhiplexpy.MatrixSolveResult) → int`

Получает количество обработанных узлов дерева решений

return

количество обработанных узлов

`get_objective_value(self: arhiplexpy.MatrixSolveResult) → float`

Получает значение целевой функции

return

значение целевой функции

`get_reduced_costs_vector(self: arhiplexpy.MatrixSolveResult) →`

`numpy.ndarray[numpy.float64]`

Получает reduced costs вектор

return

вектор reduced costs

`get_relative_gap(self: arhiplexpy.MatrixSolveResult) → float`

Получает точность решения в процентах

return

точность решения в процентах

`get_solution_status(self: arhiplexpy.MatrixSolveResult) → arhiplexpy.solution_status`

Получает статус модели в итоге расчета

return

значение статуса

`get_solution_vector(self: arhiplexpy.MatrixSolveResult) →`

`numpy.ndarray[numpy.float64]`

Получает вектор значений переменных

return

вектор переменных

`get_solve_result(self: arhiplexpy.MatrixSolveResult) → arhiplexpy.solve_result`

Получает статус процесса расчетов

return

значение статуса

`get_solve_time(self: arhiplexpy.MatrixSolveResult) → float`

Получает общее время решения

return

общее время решения [сек]

`write_solution(self: arhiplexpy.MatrixSolveResult, file_name: str) → None`

Записывает решение в файл

param file_name

путь к файлу решения для записи

`class arhiplexpy.Model`

Создает экземпляр модели

`add_constraint(*args, **kwargs)`

Overloaded function.

1. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.LinearExpression, sense: arhiplexpy.constraint_sense, rhs: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param expr

выражение для ограничения

param sense

тип ограничения (`<=`, `>=`, `==`)

param rhs

константное значение в правой части ограничения

param constr_name

имя ограничения

return

объект нового ограничения

2. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.QuadExpression, sense: arhiplexpy.constraint_sense, rhs: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param expr

выражение для ограничения

param sense

тип ограничения (`<=`, `>=` , `==`)

param rhs

константное значение в правой части ограничения

param constr_name

имя ограничения

return

объект нового ограничения

3. `add_constraint(self: arhiplexpy.Model, lhs: arhiplexpy.LinearExpression, sense: arhiplexpy.constraint_sense, rhs: arhiplexpy.LinearExpression, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param lhs

выражение для ограничения (левая часть)

param sense

переменная знака ограничения (`<=`, `>=` , `==`)

param rhs

константное значение в правой части ограничения

param constr_name

имя ограничения

return

объект нового ограничения

4. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.LinearExpression, lower_bound: float, upper_bound: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет интервальное ограничение в модель с заданными параметрами, вида `lhs <= expr <= rhs`

param expr

выражение для ограничения

param lower_bound

нижняя граница ограничения

param upper_bound

верхняя граница ограничения

param constr_name

имя ограничения

return

объект нового ограничения

```
5. add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.QuadExpression,
lower_bound: float, upper_bound: float, constr_name: str) ->
arhiplexpy.Constraint
```

Добавляет интервальное ограничение в модель с заданными параметрами, вида
 $lhs \leq expr \leq rhs$

param expr

выражение для ограничения

param lower_bound

нижняя граница ограничения

param upper_bound

верхняя граница ограничения

param constr_name

имя ограничения

return

объект нового ограничения

```
6. add_constraint(self: arhiplexpy.Model, constr: arhiplexpy.Constraint,
constr_name: str) -> arhiplexpy.Constraint
```

Добавляет уже существующее ограничение в модель

param constr

выражение для ограничения

param constr_name

имя ограничения

return

объект нового ограничения

`add_constraints(self: arhiplexpy.Model, constraints: dict) → None`

Добавляет ограничения в модель

param constraints

объект dictionary(name -> constraint)

`add_general_constraint(*args, **kwargs)`

Overloaded function.

```
1. add_general_constraint(self: arhiplexpy.Model, expr:
arhiplexpy.GeneralExpression, var: arhiplexpy.Variable, name: str) ->
arhiplexpy.GeneralConstraint
```

Добавляет общее ограничение

param expr
общее выражение

param var
переменная

param name
имя ограничения

return
объект ограничения

```
2. add_general_constraint(self: arhiplexpy.Model, expr:
    arhiplexpy.QuadExpression, var: arhiplexpy.Variable, name: str) ->
    arhiplexpy.GeneralConstraint
```

Добавляет ограничение

param expr
квадратичное выражение

param var
переменная

param name
имя ограничения

return
объект ограничения

```
3. add_general_constraint(self: arhiplexpy.Model, expr:
    arhiplexpy.LinearExpression, var: arhiplexpy.Variable, name: str) ->
    arhiplexpy.GeneralConstraint
```

Добавляет ограничение

param expr
линейное выражение

param var
переменная

param name
имя ограничения

return
объект ограничения

```
4. add_general_constraint(self: arhiplexpy.Model, gen_constr:
    arhiplexpy.GeneralConstraint) -> None
```

Добавляет общее ограничение

param gen_constr
общее ограничение

return
объект ограничения

`add_mip_start_values(*args, **kwargs)`

Overloaded function.

1. `add_mip_start_values(self: arhiplexpy.Model, start_values: dict) -> None`

Задать начальные значения переменных в решении в качестве «подсказки». Начальные значения очищаются после начала процесса решения.

param start_values

объект словаря (имя : значение)

2. `add_mip_start_values(self: arhiplexpy.Model, sol_file: str) -> None`

Задать начальные значения переменных в решении, указав файл решения

param sol_file

путь к файлу решения

`add_variable(self: arhiplexpy.Model, lower_bound: float = 0.0, upper_bound: float = inf, obj_value: float = 0.0, var_type: arhiplexpy.variable_type = <variable_type.continuous: 0>, var_name: str = '') → arhiplexpy.Variable`

Добавляет переменную в модель с заданными параметрами

param lower_bound

нижняя граница переменной

param upper_bound

верхняя граница переменной

param obj_value

коэффициент к переменной в целевой функции

param var_type

тип переменной

param var_name

имя переменной

return

объект новой переменной

`binary_var_list(self: arhiplexpy.Model, indices: list, name: str) → list`

Добавляет лист бинарных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param name

префикс имени каждой добавленной переменной

return

список переменных

`clear(self: arhiplexpy.Model) → None`

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

`clear_mip_start_values(self: arhiplexpy.Model) → None`

Удалить все стартовые значения для поиска решения

`continuous_var_list(*args, **kwargs)`

Overloaded function.

1. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: list, ub: list, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

2. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: float = inf, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

3. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: list, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

4. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: list, ub: float = inf, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

`get_bool_param(self: arhiplexpy.Model, param_name: str) -> bool`

Получает логический параметр

param param_name

имя параметра

return

значение параметра

`get_calc_uid(self: arhiplexpy.Model) -> str`

Получить уникальный идентификатор последнего удаленного расчета. :return: строку с уникальным идентификатором

`get_constraint(*args, **kwargs)`

Overloaded function.

1. `get_constraint(self: arhiplexpy.Model, idx: int) -> arhiplexpy.Constraint`

Получает ограничение из модели по индексу

param idx

индекс ограничения

return

объект ограничения

2. `get_constraint(self: arhiplexpy.Model, constr_name: str) -> arhiplexpy.Constraint`

Получает ограничение из модели по имени

param constr_name
имя ограничения
return
объект ограничения

`get_constraints_count(self: arhiplexpy.Model) -> int`

Получает количество ограничений в модели :return: количество ограничений в модели

`get_dbl_param(self: arhiplexpy.Model, param_name: str) -> float`

Получает параметр с плавающей точкой

param param_name
имя параметра
return
значение параметра

`get_general_constraint(*args, **kwargs)`

Overloaded function.

1. `get_general_constraint(self: arhiplexpy.Model, idx: int) -> arhiplexpy.GeneralConstraint`

Получает общее ограничение из модели по индексу

param idx
индекс ограничения
return
объект ограничения

2. `get_general_constraint(self: arhiplexpy.Model, constr_name: str) -> arhiplexpy.GeneralConstraint`

Получает общее ограничение из модели по имени

param constr_name
имя ограничения
return
объект ограничения

`get_general_constraints_count(self: arhiplexpy.Model) -> int`

Получает количество общих ограничений в модели :return: количество общих ограничений в модели

`get_int_param(self: arhiplexpy.Model, param_name: str) -> int`

Получает целочисленный параметр

param param_name
имя параметра

return

значение параметра

`get_name(self: arhiplexpy.Model) → str`

Получает имя модели

return

имя модели

`get_objective(self: arhiplexpy.Model) → arhiplexpy.LinearExpression`

Получает выражение целевой функции

return

объект выражения

`get_objective_sense(self: arhiplexpy.Model) → arhiplexpy.objective_sense`

Получает тип оптимизации целевой функции

return

тип оптимизации

`get_remote_solve_log(self: arhiplexpy.Model, arg0: str) → str`

Получает лог удалённого расчёта

:param calc_uid : идентификатор удалённого расчёта :return: лог удалённого расчёта

`get_string_param(self: arhiplexpy.Model, param_name: str) → str`

Получает строковый параметр

param param_name

имя параметра

return

значение параметра

`get_variable(*args, **kwargs)`

Overloaded function.

1. `get_variable(self: arhiplexpy.Model, idx: int) -> arhiplexpy.Variable`

Получает переменную модели по индексу

param idx

индекс переменной

return

объект переменной

2. `get_variable(self: arhiplexpy.Model, var_name: str) -> arhiplexpy.Variable`

Получает переменную из модели по имени

param var_name

имя переменной

return

объект переменной

`get_variables_count(self: arhiplexpy.Model) → int`

Получает количество переменных в модели

return

количество переменных в модели

`integer_var_list(*args, **kwargs)`

Overloaded function.

1. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: list, ub: list, name: str) -> list`

Добавляет лист целочисленных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

2. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: float = inf, name: str) -> list`

Добавляет лист целочисленных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

3. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: list, name: str) -> list`

Добавляет лист целочисленных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

4. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: list, ub: float = inf, name: str) -> list`

Добавляет лист целочисленных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

`is_mip(self: arhiplexpy.Model) -> bool`

Проверяет является ли задача целочисленной

`is_nonlinear(self: arhiplexpy.Model) -> bool`

Проверяет является ли задача нелинейной

`read(self: arhiplexpy.Model, file_name: str) -> None`

Читает модель из файла

param file_name

имя файла

`read_lp(self: arhiplexpy.Model, file_name: str) -> None`

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

param file_name

путь к файлу модели

`read_mps(self: arhiplexpy.Model, file_name: str) → None`

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

param file_name
путь к файлу модели

`remove(*args, **kwargs)`

Overloaded function.

1. `remove(self: arhiplexpy.Model, constr: arhiplexpy.Constraint) -> None`

Удаляет ограничение из модели

param constr
ограничение

2. `remove(self: arhiplexpy.Model, var: arhiplexpy.Variable) -> None`

Удаляет переменную из модели

param var
переменная

3. `remove(self: arhiplexpy.Model, constr: arhiplexpy.GeneralConstraint) -> None`

Удаляет общее ограничение из модели

param constr
общее ограничение

`set_bool_param(self: arhiplexpy.Model, param_name: str, param_value: bool) → None`

Задаёт логический параметр

param param_name
имя параметра

param param_value
значение параметра

`set_dbl_param(self: arhiplexpy.Model, param_name: str, param_value: float) → None`

Задаёт параметр с плавающей точкой

param param_name
имя параметра

param param_value
значение параметра

`set_int_param(self: arhiplexpy.Model, param_name: str, param_value: int) → None`

Задаёт целочисленный параметр

param param_name
имя параметра

param param_value
значение параметра

`set_log_callback(self: arhiplexpy.Model, callback: object) → None`

Задаёт функтор обратного вызова для вывода сообщений от солвера

param callback

функтор обратного вызова

`set_log_file(self: arhiplexpy.Model, log_file: str) → None`

Задать файл для записи лога решения

:param log_file файл лога

`set_name(self: arhiplexpy.Model, expr_name: str) → None`

Задаёт имя модели

param expr_name

имя модели

`set_objective(*args, **kwargs)`

Overloaded function.

1. `set_objective(self: arhiplexpy.Model, var: arhiplexpy.Variable, sense: arhiplexpy.objective_sense) -> None`

Задаёт переменную для оптимизации и тип

param var

переменная

param sense

тип оптимизации

2. `set_objective(self: arhiplexpy.Model, expr: arhiplexpy.LinearExpression, sense: arhiplexpy.objective_sense) -> None`

Задаёт выражение целевой функции и тип оптимизации

param expr

выражение

param sense

тип оптимизации

3. `set_objective(self: arhiplexpy.Model, expr: arhiplexpy.QuadExpression, sense: arhiplexpy.objective_sense) -> None`

Задаёт выражение целевой функции и тип оптимизации

param expr

выражение

param sense

тип оптимизации

`set_objective_offset(self: arhiplexpy.Model, offset: float) → None`

Задаёт константу в выражении целевой функции

param offset

значение константы

`set_objective_sense(self: arhiplexpy.Model, constr_name: arhiplexpy.objective_sense) → None`

Задаёт тип оптимизации целевой функции - максимизация или минимизация

param sense
тип оптимизации

`set_string_param(self: arhiplexpy.Model, param_name: str, param_value: str) → None`

Задаёт строковый параметр

param param_name
имя параметра

param param_value
значение параметра

`solve(self: arhiplexpy.Model) → arhiplexpy.SolveResult`

Стартует оптимизацию модели

return
объект с результатом оптимизации

`solve_remote(self: arhiplexpy.Model, name_mapping_file: str) → arhiplexpy.SolveResult`

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

param name_mapping_file путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных :**return** объект с результатом оптимизации

`solve_remote_async(self: arhiplexpy.Model, name_mapping_file: str) → None`

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения X_API_KEY.n

:**param name_mapping_file** путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

`write(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл. Тип будет определен по расширению

param file_name
путь к записываемому файлу

:**param name_mapping_file** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`write_lp(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл в формате LP

param file_name
путь к записываемому файлу

:param name_mapping_file путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`write_mps(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл в формате MPS

param file_name

путь к записываемому файлу

:param name_mapping_file путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`class arhiplexpy.QuadExpression`

Обеспечивает работу с нелинейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * переменная * коэффициент.

`add_expression(*args, **kwargs)`

Overloaded function.

1. `add_expression(self: arhiplexpy.QuadExpression, expr: arhiplexpy.QuadExpression, mult: float) -> None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

2. `add_expression(self: arhiplexpy.QuadExpression, expr: arhiplexpy.LinearExpression, mult: float) -> None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

`add_term(self: arhiplexpy.QuadExpression, var1: arhiplexpy.Variable, var2: arhiplexpy.Variable, coeff: float = 1.0) → None`

Добавляет элемент к выражению

param var1

переменная 1

param var2

переменная 2

param coeff

коэффициент

`copy(self: arhiplexpy.QuadExpression) → arhiplexpy.QuadExpression`

Создает копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

`empty(self: arhiplexpy.QuadExpression) → bool`

Возвращает истину, если выражение пустое

return

истина, если выражение пустое

`get_linear_part(self: arhiplexpy.QuadExpression) → arhiplexpy.LinearExpression`

Получает линейную часть выражения

return

линейная часть выражения

`get_name(self: arhiplexpy.QuadExpression) → str`

Получает имя выражения

return

имя выражения

`get_term_coeff(self: arhiplexpy.QuadExpression, ind: int) → float`

Возвращает коэффициент по индексу элемента в выражении

param ind

индекс элемента в выражении

return

значение коэффициента

`get_term_variable1(self: arhiplexpy.QuadExpression, ind: int) → arhiplexpy.Variable`

Возвращает первую переменную по индексу элемента в выражении

param ind

индекс элемента в выражении

return

объект переменной

`get_term_variable2(self: arhiplexpy.QuadExpression, ind: int) → arhiplexpy.Variable`

Возвращает вторую переменную по индексу элемента в выражении

param ind

индекс элемента в выражении

return

объект переменной

`get_terms_count(self: arhiplexpy.QuadExpression) → int`

Возвращает количество элементов в выражении

return

количество элементов в выражении

`remove_term(self: arhiplexpy.QuadExpression, idx: int) → None`

Удаляет элемент выражения по индексу

param idx

индекс удаляемого элемента в выражении

`set_name(self: arhiplexpy.QuadExpression, expr_name: str) → None`

Задаёт имя выражения

param expr_name

имя выражения

`set_term_coeff(self: arhiplexpy.QuadExpression, ind: int, value: float) → None`

Задаёт коэффициент по индексу выражения

param ind

индекс элемента в выражении

param value

значение коэффициента в элементе

`class arhiplexpy.SolveResult`

Результат расчёта

`get_best_bound(self: arhiplexpy.SolveResult) → float`

Получает значение граничной (двойственной) функции

return

значение граничной (двойственной) функции

`get_dual_value(*args, **kwargs)`

Overloaded function.

1. `get_dual_value(self: arhiplexpy.SolveResult, constraint: arhiplex::Constraint) -> float`

Получает значение dual value

param constraint

ограничение

return

значение dual value в решении

2. `get_dual_value(self: arhiplexpy.SolveResult, constraint_name: str) -> float`

Получает значение dual value по имени ограничения

param constraint_name

имя ограничения

return

значение dual value в решении

`get_expression_value(*args, **kwargs)`

Overloaded function.

1. `get_expression_value(self: arhiplexpy.SolveResult, arg0: arhiplex::QuadExpression) -> float`

Получает значение выражения

param expr
объект выражения
return
значение выражения

2. `get_expression_value(self: arhiplexpy.SolveResult, arg0: arhiplex::LinearExpression) -> float`

Получает значение выражения

param expr
объект выражения
return
значение выражения

`get_iterations_count(self: arhiplexpy.SolveResult) → int`

Получает количество итераций, проведенных в процессе расчета

return
количество итераций

`get_nodes_count(self: arhiplexpy.SolveResult) → int`

Получает количество обработанных узлов дерева решений

return
количество обработанных узлов

`get_objective_value(self: arhiplexpy.SolveResult) → float`

Получает значение целевой функции

return
значение целевой функции

`get_reduced_cost(*args, **kwargs)`

Overloaded function.

1. `get_reduced_cost(self: arhiplexpy.SolveResult, var_name: str) -> float`

Получает значение reduced cost в решении по имени

param var_name
имя переменной
return
значение reduced cost в решении

2. `get_reduced_cost(self: arhiplexpy.SolveResult, variable: arhiplex::Variable) -> float`

Получает значение reduced cost в решении

param variable
переменная

return

значение reduced cost в решении

`get_relative_gap(self: arhiplexpy.SolveResult) → float`

Получает точность решения в процентах

return

точность решения в процентах

`get_solution_status(self: arhiplexpy.SolveResult) → arhiplexpy.solution_status`

Получает статус модели в итоге расчета

return

значение статуса

`get_solve_result(self: arhiplexpy.SolveResult) → arhiplexpy.solve_result`

Получает статус процесса расчетов

return

значение статуса

`get_solve_time(self: arhiplexpy.SolveResult) → float`

Получает общее время решения

return

общее время решения [сек]

`get_variable_value(*args, **kwargs)`

Overloaded function.

1. `get_variable_value(self: arhiplexpy.SolveResult, var: arhiplex::Variable) -> float`

Получает значение переменной в решении

param var

объект переменной

return

значение переменной в решении

2. `get_variable_value(self: arhiplexpy.SolveResult, var_name: str) -> float`

Получает значение переменной в решении по имени

param var_name

имя переменной

return

значение переменной в решении

`write_solution(self: arhiplexpy.SolveResult, file_name: str) → None`

Записывает решение в файл

param file_name

путь к файлу решения для записи

`class arhiplexpy.Variable`

Работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

`get_lower_bound(self: arhiplexpy.Variable) → float`

Получает нижнюю границу переменной

return

нижнюю границу переменной

`get_name(self: arhiplexpy.Variable) → str`

Получает имя переменной в модели

return

имя переменной

`get_type(self: arhiplexpy.Variable) → arhiplexpy.variable_type`

Получает тип переменной

return

тип переменной

`get_upper_bound(self: arhiplexpy.Variable) → float`

Получает верхнюю границу переменной

return

верхнюю границу переменной

`remove(self: arhiplexpy.Variable) → None`

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

`set_lower_bound(self: arhiplexpy.Variable, lower_bound: float) → None`

Задаёт нижнюю границу для переменной

param lower_bound

новая нижняя граница переменной

`set_name(self: arhiplexpy.Variable, var_name: str) → None`

Задаёт имя переменной в модели

param var_name

имя переменной

`set_type(self: arhiplexpy.Variable, type: arhiplexpy.variable_type) → None`

Задаёт новый тип переменной в модели

param type

новый тип переменной

`set_upper_bound(self: arhiplexpy.Variable, upper_bound: float) → None`

Задаёт верхнюю границу для переменной

param upper_bound

новая верхняя граница переменной

`class arhiplexpy.constraint_sense`

Members:

`equal`

`less_equal`

`greater_equal`

property name

class arhiplexpy.gen_expr_type

Members:

constant

variable

minus

plus

divide

multiply

func_square_root

func_sqr

func_pow

func_exp

func_log

func_log2

func_log10

func_sin

func_cos

func_tan

func_logistic

func_tanh

func_signpow

property name

class arhiplexpy.objective_sense

Members:

minimize

maximize

property name

class arhiplexpy.solution_status

Members:

invalid_solution

optimal

feasible

infeasible

unbounded

infeasible_or_unbounded

property name

class `arhiplexpy.solve_result`

Members:

success

fail

remote_invalid_api_key

remote_api_key_not_set

remote_time_amount_is_over

remote_time_per_calc_violated

remote_fail

property name

`arhiplexpy.sum(iterable: Iterable) → float | arhiplexpy.Variable | arhiplexpy.LinearExpression | arhiplexpy.QuadExpression | arhiplexpy.GeneralExpression`

Суммирует переменные и/или линейные выражения

return

объект выражения

class `arhiplexpy.variable_type`

Members:

continuous

binary

integer

semicontinuous

semiinteger

property name

Параметры

Имя параметра	Описание
http_proxy	Http Proxy используемый для работы с облаком Type: string Default value:
log_display_systime	Включает или отключает вывод системного времени в лог. Неприменимо для удаленного расчета ArhiCloud. Type: bool Default value: false
log_to_console	Включает или отключает вывод солвера на консоль. Также применимо для удаленного расчета ArhiCloud. Type: bool Default value: true

continues on next page

Таблица 4.1 – продолжение с предыдущей страницы

Имя параметра	Описание
mip_abs_gap	<p>Абсолютная ошибка, $\text{abs}(\text{ub}-\text{lb})$, для определения степени оптимальности MIP задачи</p> <p>Type: floating point</p> <p>Default value: 0.00</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
mip_feasibility_tolerance	<p>Точность достижимости MIP задачи.</p> <p>Type: floating point</p> <p>Default value: 0.00</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
mip_rel_gap	<p>Относительная ошибка, $\text{abs}(\text{ub}-\text{lb})/\text{abs}(\text{ub})$, для определения степени оптимальности MIP задачи. Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: floating point</p> <p>Default value: 0.00</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
multiprocess_max_process_count	<p>Количество процессов для запуска</p> <p>Type: integral</p> <p>Default value: 0</p> <p>Lower bound: 0</p> <p>Upper bound: 16</p>

continues on next page

Таблица 4.1 – продолжение с предыдущей страницы

Имя параметра	Описание
presolve	<p>Пресолвер: «off» или «on». Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: string</p> <p>Default value: on</p>
random_seed	<p>Начальное значение для генератора случайных чисел</p> <p>Type: integral</p> <p>Default value: 0</p> <p>Lower bound: 0</p> <p>Upper bound: 2147483647</p>
remote_timeout	<p>Максимальное время ожидания сервера удаленных вычислений (ArhiCloud), сек</p> <p>Type: integral</p> <p>Default value: 300</p> <p>Lower bound: 10</p> <p>Upper bound: 600</p>
threads	<p>number of threads used by ArhiPlex (0: automatic)</p> <p>Type: integral</p> <p>Default value: 0</p> <p>Lower bound: 0</p> <p>Upper bound: 2147483647</p>

continues on next page

Таблица 4.1 – продолжение с предыдущей страницы

Имя параметра	Описание
time_limit	<p>Лимит времени в сек. Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: floating point</p> <p>Default value: inf</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
write_zeroes_to_solution	<p>Записывать нулевые значения переменных в файл решения</p> <p>Type: bool</p> <p>Default value: true</p>

a

arhiplexpy, [42](#)

A

<code>add_constant()</code>	(<i>method</i> <i>arhiplexpy.LinearExpression</i>), 45	<code>arhiplex::Constraint::GetLowerBound</code>	(<i>C++ function</i>), 2
<code>add_constraint()</code>	(<i>method</i> <i>arhiplexpy.MatrixModel</i>), 61	<code>arhiplex::Constraint::GetName</code>	(<i>C++ function</i>), 3
<code>add_constraint()</code>	(<i>method</i> <i>arhiplexpy.Model</i>), 66	<code>arhiplex::Constraint::GetQuadExpression</code>	(<i>C++ function</i>), 2
<code>add_constraints()</code>	(<i>method</i> <i>arhiplexpy.MatrixModel</i>), 62	<code>arhiplex::Constraint::GetRange</code>	(<i>C++ function</i>), 2
<code>add_constraints()</code>	(<i>method</i> <i>arhiplexpy.Model</i>), 68	<code>arhiplex::Constraint::GetUpperBound</code>	(<i>C++ function</i>), 2
<code>add_expr()</code>	(<i>method</i> <i>arhiplexpy.GeneralExpression</i>), 44	<code>arhiplex::Constraint::Remove</code>	(<i>C++ function</i>), 2
<code>add_expression()</code>	(<i>method</i> <i>arhiplexpy.LinearExpression</i>), 45	<code>arhiplex::Constraint::SetLowerBound</code>	(<i>C++ function</i>), 2
<code>add_expression()</code>	(<i>method</i> <i>arhiplexpy.QuadExpression</i>), 80	<code>arhiplex::Constraint::SetName</code>	(<i>C++ function</i>), 3
<code>add_general_constraint()</code>	(<i>method</i> <i>arhiplexpy.Model</i>), 68	<code>arhiplex::Constraint::SetRange</code>	(<i>C++ function</i>), 2
<code>add_mip_start_values()</code>	(<i>method</i> <i>arhiplexpy.Model</i>), 69	<code>arhiplex::Constraint::SetUpperBound</code>	(<i>C++ function</i>), 2
<code>add_term()</code>	(<i>method</i> <i>arhiplexpy.LinearExpression</i>), 45	<code>arhiplex::constraint_sense</code>	(<i>C++ enum</i>), 19
<code>add_term()</code>	(<i>method</i> <i>arhiplexpy.QuadExpression</i>), 80	<code>arhiplex::constraint_sense::equal</code>	(<i>C++ enumerator</i>), 19
<code>add_variable()</code>	(<i>method</i> <i>arhiplexpy.Model</i>), 70	<code>arhiplex::constraint_sense::greater_equal</code>	(<i>C++ enumerator</i>), 19
<code>add_variables()</code>	(<i>method</i> <i>arhiplexpy.MatrixModel</i>), 62	<code>arhiplex::constraint_sense::less_equal</code>	(<i>C++ enumerator</i>), 19
<code>arhiplex::arhiplex_exception</code>	(<i>C++ class</i>), 1	<code>arhiplex::LinearExpression</code>	(<i>C++ class</i>), 3
<code>arhiplex::arhiplex_exception::get_error_code</code>	(<i>C++ function</i>), 1	<code>arhiplex::LinearExpression::AddConstant</code>	(<i>C++ function</i>), 4
<code>arhiplex::Constraint</code>	(<i>C++ class</i>), 1	<code>arhiplex::LinearExpression::AddExpression</code>	(<i>C++ function</i>), 5
<code>arhiplex::Constraint::Constraint</code>	(<i>C++ function</i>), 1	<code>arhiplex::LinearExpression::AddTerm</code>	(<i>C++ function</i>), 4
<code>arhiplex::Constraint::GetLinearExpression</code>	(<i>C++ function</i>), 2	<code>arhiplex::LinearExpression::CreateFreeCopy</code>	(<i>C++ function</i>), 5
		<code>arhiplex::LinearExpression::empty</code>	(<i>C++ function</i>), 3

arhiplex::LinearExpression::GetConstant	15		
(C++ function), 4		arhiplex::Model::GetObjective	(C++ function), 13
arhiplex::LinearExpression::GetName		arhiplex::Model::GetObjectiveSense	(C++ function), 12
(C++ function), 3		arhiplex::Model::GetRemoteSolveLog	(C++ function), 14
arhiplex::LinearExpression::GetTermCoeff		arhiplex::Model::GetStringParam	(C++ function), 8
(C++ function), 4		arhiplex::Model::GetVariable	(C++ function), 11
arhiplex::LinearExpression::GetTermsCount		arhiplex::Model::GetVariableByName	(C++ function), 12
(C++ function), 3		arhiplex::Model::GetVariablesCount	(C++ function), 12
arhiplex::LinearExpression::GetTermVariable		arhiplex::Model::IsMip	(C++ function), 15
(C++ function), 4		arhiplex::Model::IsNonlinear	(C++ function), 15
arhiplex::LinearExpression::GetTermVariableIndex	function), 11	arhiplex::Model::Model	(C++ function), 7
(C++ function), 4		arhiplex::Model::Read	(C++ function), 7
arhiplex::LinearExpression::LinearExpression	(C++ function), 12	arhiplex::Model::ReadLp	(C++ function), 8
(C++ function), 3		arhiplex::Model::ReadMps	(C++ function), 8
arhiplex::LinearExpression::RemoveTerm	(C++ function), 12	arhiplex::Model::Remove	(C++ function), 14
(C++ function), 5		arhiplex::Model::SetBoolParam	(C++ function), 9
arhiplex::LinearExpression::RemoveVariable	(C++ function), 5	arhiplex::Model::SetDblParam	(C++ function), 9
(C++ function), 5		arhiplex::Model::SetIntParam	(C++ function), 8
arhiplex::LinearExpression::SetConstant	(C++ function), 4	arhiplex::Model::SetLogCallback	(C++ function), 14
(C++ function), 4		arhiplex::Model::SetLogFile	(C++ function), 14
arhiplex::LinearExpression::SetName	(C++ function), 3	arhiplex::Model::SetName	(C++ function), 15
(C++ function), 3		arhiplex::Model::SetObjective	(C++ function), 13
arhiplex::LinearExpression::SetTermCoeff	(C++ function), 4	arhiplex::Model::SetObjectiveOffset	(C++ function), 12
(C++ function), 4		arhiplex::Model::SetObjectiveSense	(C++ function), 12
arhiplex::Model (C++ class), 7		arhiplex::Model::SetStringParam	(C++ function), 9
arhiplex::Model::AddConstraint	(C++ function), 10, 11	arhiplex::Model::Solve	(C++ function), 13
arhiplex::Model::AddMipStartValue	(C++ function), 14	arhiplex::Model::SolveRemote	(C++ function), 13
arhiplex::Model::AddMipStartValues	(C++ function), 14	arhiplex::Model::SolveRemoteAsync	(C++ function), 13
arhiplex::Model::AddVariable	(C++ function), 9	arhiplex::Model::Write	(C++ function), 9
arhiplex::Model::Clear	(C++ function), 14		
arhiplex::Model::ClearMipStartValues	(C++ function), 15		
arhiplex::Model::GetBoolParam	(C++ function), 8		
arhiplex::Model::GetCalcUID	(C++ function), 15		
arhiplex::Model::GetConstraint	(C++ function), 12		
arhiplex::Model::GetConstraintsCount	(C++ function), 12		
arhiplex::Model::GetConstrByName	(C++ function), 12		
arhiplex::Model::GetDblParam	(C++ function), 8		
arhiplex::Model::GetIntParam	(C++ function), 8		
arhiplex::Model::GetName	(C++ function),		

arhiplex::Model::WriteLp (C++ function), 9	arhiplex::solve_result (C++ enum), 20
arhiplex::Model::WriteMps (C++ function), 9	arhiplex::solve_result::fail (C++ enumerator), 20
arhiplex::objective_sense (C++ enum), 19	arhiplex::solve_result::remote_api_key_not_set (C++ enumerator), 20
arhiplex::objective_sense::maximize (C++ enumerator), 19	arhiplex::solve_result::remote_fail (C++ enumerator), 20
arhiplex::objective_sense::minimize (C++ enumerator), 19	arhiplex::solve_result::remote_invalid_api_key (C++ enumerator), 20
arhiplex::QuadExpression (C++ class), 5	arhiplex::solve_result::remote_time_amount_is_over (C++ enumerator), 20
arhiplex::QuadExpression::AddExpression (C++ function), 7	arhiplex::solve_result::remote_time_per_calc_violate (C++ enumerator), 20
arhiplex::QuadExpression::AddTerm (C++ function), 6	arhiplex::solve_result::success (C++ enumerator), 20
arhiplex::QuadExpression::CreateFreeCopy (C++ function), 7	arhiplex::SolveResult (C++ class), 15
arhiplex::QuadExpression::empty (C++ function), 6	arhiplex::SolveResult::GetBestBoundValue (C++ function), 16
arhiplex::QuadExpression::GetName (C++ function), 6	arhiplex::SolveResult::GetDualValue (C++ function), 16, 17
arhiplex::QuadExpression::GetTermCoeff (C++ function), 6	arhiplex::SolveResult::GetExpressionValue (C++ function), 17
arhiplex::QuadExpression::GetTermsCount (C++ function), 5	arhiplex::SolveResult::GetIterationsCount (C++ function), 16
arhiplex::QuadExpression::GetTermVariable1 (C++ function), 6	arhiplex::SolveResult::GetObjectiveFunctionValue (C++ function), 16
arhiplex::QuadExpression::GetTermVariable2 (C++ function), 6	arhiplex::SolveResult::GetProcessedNodesCount (C++ function), 16
arhiplex::QuadExpression::QuadExpression (C++ function), 5	arhiplex::SolveResult::GetReducedCost (C++ function), 17
arhiplex::QuadExpression::RemoveTerm (C++ function), 7	arhiplex::SolveResult::GetRelativeGap (C++ function), 16
arhiplex::QuadExpression::SetName (C++ function), 6	arhiplex::SolveResult::GetSolutionStatus (C++ function), 15
arhiplex::QuadExpression::SetTermCoeff (C++ function), 6	arhiplex::SolveResult::GetSolveResult (C++ function), 15
arhiplex::solution_status (C++ enum), 20	arhiplex::SolveResult::GetSolveTime (C++ function), 16
arhiplex::solution_status::feasible (C++ enumerator), 20	arhiplex::SolveResult::GetVariableValue (C++ function), 16
arhiplex::solution_status::infeasible (C++ enumerator), 20	arhiplex::SolveResult::WriteSolution (C++ function), 17
arhiplex::solution_status::infeasible_or_unbounded (C++ enumerator), 21	arhiplex::Variable (C++ class), 17
arhiplex::solution_status::invalid_solution (C++ enumerator), 20	arhiplex::Variable::GetLowerBound (C++ function), 18
arhiplex::solution_status::optimal (C++ enumerator), 20	arhiplex::Variable::GetName (C++ function), 18
arhiplex::solution_status::unbounded (C++ enumerator), 21	arhiplex::Variable::GetType (C++ function), 18
	arhiplex::Variable::GetUpperBound (C++ function), 18

<code>arhiplex::Variable::Remove</code>	(C++ function), 18	G	<code>gen_expr_type</code> (класс в <i>arhiplexpy</i>), 86
<code>arhiplex::Variable::SetLowerBound</code>	(C++ function), 18	<code>GeneralConstraint</code> (класс в <i>arhiplexpy</i>), 43	<code>GeneralExpression</code> (класс в <i>arhiplexpy</i>), 44
<code>arhiplex::Variable::SetName</code>	(C++ function), 18	<code>get_arity()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>arhiplex::Variable::SetType</code>	(C++ function), 18	<code>get_best_bound()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65
<code>arhiplex::Variable::SetUpperBound</code>	(C++ function), 18	<code>get_best_bound()</code>	(метод <i>arhiplexpy.SolveResult</i>), 82
<code>arhiplex::variable_type</code> (C++ enum), 19		<code>get_bool_param()</code>	(метод <i>arhiplexpy.MatrixModel</i>), 62
<code>arhiplex::variable_type::binary</code>	(C++ enumerator), 19	<code>get_bool_param()</code>	(метод <i>arhiplexpy.Model</i>), 72
<code>arhiplex::variable_type::continuous</code>	(C++ enumerator), 19	<code>get_calc_uid()</code>	(метод <i>arhiplexpy.Model</i>), 72
<code>arhiplex::variable_type::integer</code>	(C++ enumerator), 19	<code>get_constant()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>arhiplex::variable_type::semicontinuous</code>	(C++ enumerator), 19	<code>get_constant()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 45
<code>arhiplex::variable_type::semiinteger</code>	(C++ enumerator), 19	<code>get_constraint()</code>	(метод <i>arhiplexpy.Model</i>), 72
<code>ArhiplexException</code> , 42		<code>get_constraints_count()</code>	(метод <i>arhiplexpy.Model</i>), 73
<code>arhiplexpy</code>	module, 42	<code>get_dbl_param()</code>	(метод <i>arhiplexpy.MatrixModel</i>), 63
B		<code>get_dbl_param()</code>	(метод <i>arhiplexpy.Model</i>), 73
<code>binary_var_list()</code>	(метод <i>arhiplexpy.Model</i>), 70	<code>get_dual_value()</code>	(метод <i>arhiplexpy.SolveResult</i>), 82
C		<code>get_dual_values_vector()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65
<code>clear()</code> (метод <i>arhiplexpy.MatrixModel</i>), 62		<code>get_expr()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>clear()</code> (метод <i>arhiplexpy.Model</i>), 70		<code>get_expr_count()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>clear_mip_start_values()</code>	(метод <i>arhiplexpy.Model</i>), 70	<code>get_expression()</code>	(метод <i>arhiplexpy.GeneralConstraint</i>), 43
<code>Constraint</code> (класс в <i>arhiplexpy</i>), 42		<code>get_expression_value()</code>	(метод <i>arhiplexpy.SolveResult</i>), 82
<code>constraint_sense</code> (класс в <i>arhiplexpy</i>), 85		<code>get_general_constraint()</code>	(метод <i>arhiplexpy.Model</i>), 73
<code>continuous_var_list()</code>	(метод <i>arhiplexpy.Model</i>), 71	<code>get_general_constraints_count()</code>	(метод <i>arhiplexpy.Model</i>), 73
<code>copy()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44	<code>get_int_param()</code>	(метод <i>arhiplexpy.MatrixModel</i>), 63
<code>copy()</code> (метод <i>arhiplexpy.LinearExpression</i>), 45		<code>get_int_param()</code>	(метод <i>arhiplexpy.Model</i>), 73
<code>copy()</code> (метод <i>arhiplexpy.QuadExpression</i>), 80		<code>get_iterations_count()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65
<code>cos()</code> (метод <i>arhiplexpy.MathFunc</i>), 47		<code>get_iterations_count()</code>	(метод
E			
<code>empty()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 45		
<code>empty()</code> (метод <i>arhiplexpy.QuadExpression</i>), 81			
<code>exp()</code> (метод <i>arhiplexpy.MathFunc</i>), 48			

<i>arhiplexpy.SolveResult</i>), 83		<i>arhiplexpy.MatrixSolveResult</i>), 66	
<code>get_linear_expression()</code>	(метод <i>arhiplexpy.Constraint</i>), 42	<code>get_solve_result()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 66
<code>get_linear_part()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81	<code>get_solve_result()</code>	(метод <i>arhiplexpy.SolveResult</i>), 84
<code>get_lower_bound()</code>	(метод <i>arhiplexpy.Constraint</i>), 42	<code>get_solve_time()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 66
<code>get_lower_bound()</code>	(метод <i>arhiplexpy.Variable</i>), 84	<code>get_solve_time()</code>	(метод <i>arhiplexpy.SolveResult</i>), 84
<code>get_name()</code> (метод <i>arhiplexpy.Constraint</i>), 42		<code>get_string_param()</code>	(метод <i>arhiplexpy.MatrixModel</i>), 63
<code>get_name()</code>	(метод <i>arhiplexpy.GeneralConstraint</i>), 43	<code>get_string_param()</code>	(метод <i>arhiplexpy.Model</i>), 74
<code>get_name()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 45	<code>get_term_coeff()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 46
<code>get_name()</code> (метод <i>arhiplexpy.Model</i>), 74		<code>get_term_coeff()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81
<code>get_name()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81	<code>get_term_variable()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 46
<code>get_name()</code> (метод <i>arhiplexpy.Variable</i>), 85		<code>get_term_variable1()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81
<code>get_nodes_count()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65	<code>get_term_variable2()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81
<code>get_nodes_count()</code>	(метод <i>arhiplexpy.SolveResult</i>), 83	<code>get_term_variable_idx()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 46
<code>get_objective()</code> (метод <i>arhiplexpy.Model</i>), 74		<code>get_terms_count()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 46
<code>get_objective_sense()</code>	(метод <i>arhiplexpy.Model</i>), 74	<code>get_terms_count()</code>	(метод <i>arhiplexpy.QuadExpression</i>), 81
<code>get_objective_value()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65	<code>get_type()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>get_objective_value()</code>	(метод <i>arhiplexpy.SolveResult</i>), 83	<code>get_type()</code> (метод <i>arhiplexpy.Variable</i>), 85	
<code>get_quad_expression()</code>	(метод <i>arhiplexpy.Constraint</i>), 42	<code>get_upper_bound()</code>	(метод <i>arhiplexpy.Constraint</i>), 42
<code>get_quad_part()</code>	(метод <i>arhiplexpy.LinearExpression</i>), 46	<code>get_upper_bound()</code>	(метод <i>arhiplexpy.Variable</i>), 85
<code>get_range()</code> (метод <i>arhiplexpy.Constraint</i>), 42		<code>get_variable()</code>	(метод <i>arhiplexpy.GeneralConstraint</i>), 43
<code>get_reduced_cost()</code>	(метод <i>arhiplexpy.SolveResult</i>), 83	<code>get_variable()</code>	(метод <i>arhiplexpy.GeneralExpression</i>), 44
<code>get_reduced_costs_vector()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65	<code>get_variable()</code> (метод <i>arhiplexpy.Model</i>), 74	
<code>get_relative_gap()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65	<code>get_variable_value()</code>	(метод <i>arhiplexpy.SolveResult</i>), 84
<code>get_relative_gap()</code>	(метод <i>arhiplexpy.SolveResult</i>), 84	<code>get_variables_count()</code>	(метод <i>arhiplexpy.MatrixModel</i>), 63
<code>get_remote_solve_log()</code>	(метод <i>arhiplexpy.Model</i>), 74	<code>get_variables_count()</code>	(метод <i>arhiplexpy.Model</i>), 74
<code>get_solution_status()</code>	(метод <i>arhiplexpy.MatrixSolveResult</i>), 65		
<code>get_solution_status()</code>	(метод <i>arhiplexpy.SolveResult</i>), 84	<code>integer_var_list()</code>	(метод
<code>get_solution_vector()</code>	(метод		

- arhiplexpy.Model*), 75
- is_mip()* (метод *arhiplexpy.Model*), 76
- is_nonlinear()* (метод *arhiplexpy.Model*), 76
- ## L
- LinearExpression* (класс в *arhiplexpy*), 45
- log()* (метод *arhiplexpy.MathFunc*), 48
- log10()* (метод *arhiplexpy.MathFunc*), 49
- log2()* (метод *arhiplexpy.MathFunc*), 50
- logistic()* (метод *arhiplexpy.MathFunc*), 50
- ## M
- MathFunc* (класс в *arhiplexpy*), 47
- MatrixModel* (класс в *arhiplexpy*), 61
- MatrixSolveResult* (класс в *arhiplexpy*), 65
- Model* (класс в *arhiplexpy*), 66
- module*
 arhiplexpy, 42
- ## N
- name* (*arhiplexpy.constraint_sense* property), 85
- name* (*arhiplexpy.gen_expr_type* property), 86
- name* (*arhiplexpy.objective_sense* property), 86
- name* (*arhiplexpy.solution_status* property), 87
- name* (*arhiplexpy.solve_result* property), 87
- name* (*arhiplexpy.variable_type* property), 87
- ## O
- objective_sense* (класс в *arhiplexpy*), 86
- ## P
- pow()* (метод *arhiplexpy.MathFunc*), 51
- ## Q
- QuadExpression* (класс в *arhiplexpy*), 80
- ## R
- read()* (метод *arhiplexpy.Model*), 76
- read_lp()* (метод *arhiplexpy.Model*), 76
- read_mps()* (метод *arhiplexpy.Model*), 76
- remove()* (метод *arhiplexpy.Constraint*), 43
- remove()* (метод *arhiplexpy.GeneralConstraint*), 43
- remove()* (метод *arhiplexpy.Model*), 77
- remove()* (метод *arhiplexpy.Variable*), 85
- remove_term()* (метод *arhiplexpy.LinearExpression*), 46
- remove_term()* (метод *arhiplexpy.QuadExpression*), 81
- remove_variable()* (метод *arhiplexpy.LinearExpression*), 46
- ## S
- set_bool_param()* (метод *arhiplexpy.MatrixModel*), 63
- set_bool_param()* (метод *arhiplexpy.Model*), 77
- set_constant()* (метод *arhiplexpy.LinearExpression*), 46
- set_dbl_param()* (метод *arhiplexpy.MatrixModel*), 63
- set_dbl_param()* (метод *arhiplexpy.Model*), 77
- set_expr()* (метод *arhiplexpy.GeneralExpression*), 45
- set_int_param()* (метод *arhiplexpy.MatrixModel*), 63
- set_int_param()* (метод *arhiplexpy.Model*), 77
- set_log_callback()* (метод *arhiplexpy.Model*), 77
- set_log_file()* (метод *arhiplexpy.MatrixModel*), 64
- set_log_file()* (метод *arhiplexpy.Model*), 78
- set_lower_bound()* (метод *arhiplexpy.Constraint*), 43
- set_lower_bound()* (метод *arhiplexpy.Variable*), 85
- set_name()* (метод *arhiplexpy.Constraint*), 43
- set_name()* (метод *arhiplexpy.GeneralConstraint*), 44
- set_name()* (метод *arhiplexpy.LinearExpression*), 47
- set_name()* (метод *arhiplexpy.Model*), 78
- set_name()* (метод *arhiplexpy.QuadExpression*), 82
- set_name()* (метод *arhiplexpy.Variable*), 85
- set_objective()* (метод *arhiplexpy.MatrixModel*), 64
- set_objective()* (метод *arhiplexpy.Model*), 78
- set_objective_offset()* (метод *arhiplexpy.MatrixModel*), 64
- set_objective_offset()* (метод *arhiplexpy.Model*), 78
- set_objective_sense()* (метод *arhiplexpy.Model*), 78
- set_range()* (метод *arhiplexpy.Constraint*), 43

`set_string_param()` (метод *arhiplexpy.MatrixModel*), 64
`set_string_param()` (метод *arhiplexpy.Model*), 79
`set_term_coeff()` (метод *arhiplexpy.LinearExpression*), 47
`set_term_coeff()` (метод *arhiplexpy.QuadExpression*), 82
`set_type()` (метод *arhiplexpy.Variable*), 85
`set_upper_bound()` (метод *arhiplexpy.Constraint*), 43
`set_upper_bound()` (метод *arhiplexpy.Variable*), 85
`signpow()` (метод *arhiplexpy.MathFunc*), 54
`sin()` (метод *arhiplexpy.MathFunc*), 58
`solution_status` (класс в *arhiplexpy*), 86
`solve()` (метод *arhiplexpy.MatrixModel*), 64
`solve()` (метод *arhiplexpy.Model*), 79
`solve_remote()` (метод *arhiplexpy.MatrixModel*), 64
`solve_remote()` (метод *arhiplexpy.Model*), 79
`solve_remote_async()` (метод *arhiplexpy.Model*), 79
`solve_result` (класс в *arhiplexpy*), 87
`SolveResult` (класс в *arhiplexpy*), 82
`sqr()` (метод *arhiplexpy.MathFunc*), 59
`sqrt()` (метод *arhiplexpy.MathFunc*), 59
`sum()` (в модуле *arhiplexpy*), 87

T

`tan()` (метод *arhiplexpy.MathFunc*), 60
`tanh()` (метод *arhiplexpy.MathFunc*), 61

V

`Variable` (класс в *arhiplexpy*), 84
`variable_type` (класс в *arhiplexpy*), 87

W

`write()` (метод *arhiplexpy.MatrixModel*), 64
`write()` (метод *arhiplexpy.Model*), 79
`write_lp()` (метод *arhiplexpy.MatrixModel*), 64
`write_lp()` (метод *arhiplexpy.Model*), 79
`write_mps()` (метод *arhiplexpy.MatrixModel*), 65
`write_mps()` (метод *arhiplexpy.Model*), 80
`write_solution()` (метод *arhiplexpy.MatrixSolveResult*), 66
`write_solution()` (метод *arhiplexpy.SolveResult*), 84