ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Программирование на языке Python. Уровень 1.

Модули и пакеты.

Модули в Python

- Модулем в языке Python называется любой файл с программным кодом.
- Каждый модуль может импортировать другой модуль, получая таким образом доступ к **атрибутам** (переменным, функциям и классам), объявленным внутри импортированного модуля.
- Импортируемый модуль может содержать программу не только на **Python** – можно импортировать скомпилированный модуль, написанный на языке **C**.
- Все программы, которые мы запускали ранее, были расположены в модуле с названием "__main__".
- Получить имя модуля позволяет предопределенный атрибут __name__.

```
print(__name__)  # __main__ или <имя_модуля>
```

- Для запускаемого модуля атрибут __name__ содержит значение "__main__", а для импортируемого модуля – его имя.

Импорт модулей. Инструкция import

- Импортировать модуль позволяет инструкция **import.**
- Инструкция import имеет следующий формат:

```
import <Название модуля 1> [as <Псевдоним 1>][, ...,
       <Название модуля N> [as <Псевдоним N>]]
```

- После ключевого слова import указывается название модуля.
- Название не должно содержать расширения и пути к файлу. При именовании модулей необходимо учитывать, что операция импорта создает одноименный идентификатор.
- Название модуля должно полностью соответствовать правилам именования переменных.
- Допускается создавать модуль с именем, начинающимся с цифры, но импортировать такой модуль нельзя.
- Следует избегать совпадения имен модулей с ключевыми словами, встроенными идентификаторами и названиями модулей, входящих в стандартную библиотеку.
- За один раз можно импортировать сразу несколько модулей, записав их через запятую.

Импорт модулей. Инструкция from

- Для импортирования только определенных идентификаторов из модуля предназначается инструкция from.
- Формат инструкции from:

```
from <Название модуля> import <Идентификатор1> [as <Псевдоним1>]
[,..., <Идентификатор1> [as <Псевдоним1>]
```

- Пример.

```
from math import pi, floor as f
print(pi)                  # Вывод числа pi
# Вызываем функцию floor() через идентификатор f
print(f(5.49))             # Выведет: 5
```

- Формат инструкции from для импорта всех идентификаторов:

```
from <Название модуля> import *
```

- Пример.

```
from math import * # Импортируем все идентификаторы из модуля math
```

Пакеты

- Пакетом называется каталог с модулями, в котором расположен файл инициализации __init__.py.
- Файл инициализации может быть пустым или содержать код, который будет выполнен при первой операции импортирования любого модуля, входящего в состав пакета.
- Файл инициализации должен присутствовать внутри каталога с модулями. *Примечание. Начиная с версии Python 3.3, добавлять файл `__init__.py` в папку с модулями не обязательно, интерпретатор Python считает все папки пакетами.*
- Пакеты позволяют распределить модули по каталогам.
- Чтобы импортировать модуль, расположенный во вложенном каталоге, необходимо указать путь к нему, задав имена каталогов через точку.

- Пример.
- Если модуль расположен в каталоге C:\folder1\folder2\, то путь к нему из C:\ должен быть записан так: folder1.folder2.

- При использовании инструкции import путь к модулю должен включать не только имена каталогов, но и название модуля без расширения:

```
import folder1.folder2.module
```

Менеджер пакетов PIP

- PIP (рекурсивная аббревиатура от «Pip Installs Packages» или «Pip Installs Python») – стандартный кроссплатформенный менеджер пакетов в Python.
- PIP поставляется вместе с Python, и доступен после его установки.
- Проверить установку PIP, а также узнать номер установленной версии можно следующей командой:

  `pip --version`
- Если по какой-то причине PIP не установлен в системе, то существует возможность дополнительной установки.
- Пример. Установка в Windows:

  1. Скачать файл get-pip.py.
  2. Открыть командную строку и перейти в папку, в которой сохранен get-pip.py.
  3. В командной строке выполнить команду: **`python get-pip.py`**
  4. Установка PIP завершена!

Работа с PIP

- Синтаксис работы с PIP:

  pip + команда + доп. опции

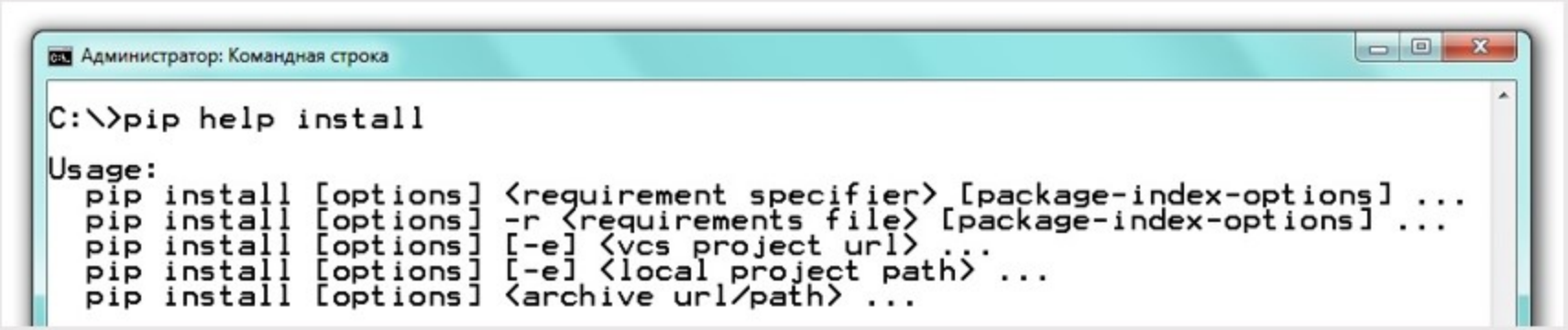
  `pip <command> [options]`

- Со всеми командами PIP можно ознакомиться, выполнив команду:

  `pip help`

- Информация по конкретной команде:

  `pip help <command>`

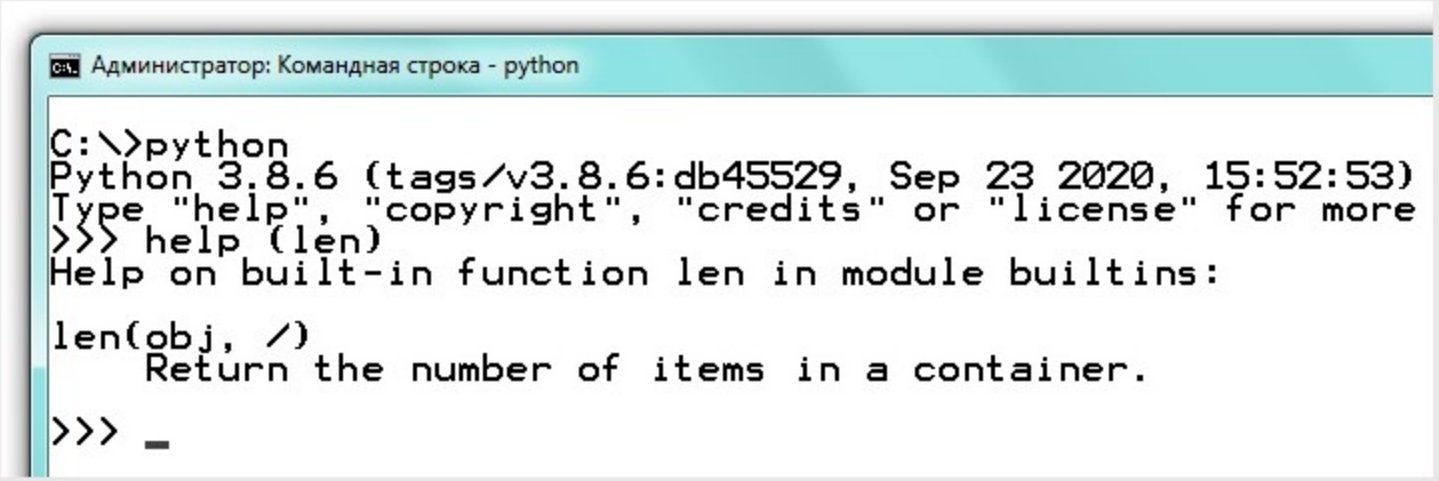


Команды PIP

- pip install package-name - устанавливает последнюю версию пакета;
- pip install package-name==4.8.2 - устанавливает пакет версии 4.8.2;
- pip install package-name --upgrade - обновляет версию пакета;
- pip download - скачивает пакеты;
- pip uninstall - удаляет пакеты;
- pip freeze - выводит список установленных пакетов в необходимом формате (обычно используется для записи в requirements.txt);
- pip list - выводит список установленных пакетов;
- pip list --outdated - выводит список устаревших пакетов;
- pip show - показывает информацию об установленном пакете;
- pip check - проверяет установленные пакеты на совместимость зависимостей;
- pip search - по введенному названию, ищет пакеты, опубликованные в PyPI;
- pip wheel - собирает wheel-архив по вашим требованиям и зависимостям;
- pip hash - вычисляет хеши архивов пакетов;
- pip completion - вспомогательная команда используется для завершения основной команды;
- pip help - помощь по командам.

Обзор стандартной библиотеки

- Одним из важных преимуществ языка Python является наличие большой библиотеки модулей и пакетов, входящих в стандартную поставку.
- В среде Python без дополнительных операций импорта доступно более сотни встроенных объектов, в основном, функций и исключений.
- Для удобства функции условно разделены по категориям.
- Уточнить назначение функции, ее аргументов и результата можно в интерактивной сессии интерпретатора Python:



Встроенные функции

- 1. Функции преобразования типов и классы: `coerce`, `str`, `repr`, `int`, `list`, `tuple`, `long`, `float`, `complex`, `dict`, `super`, `file`, `bool`, `object`.
- 2. Числовые и строковые функции: `abs`, `divmod`, `ord`, `pow`, `len`, `chr`, `unichr`, `hex`, `oct`, `cmp`, `round`, `unicode`.
- 3. Функции обработки данных: `apply`, `map`, `filter`, `reduce`, `zip`, `range`, `xrange`, `max`, `min`, `iter`, `enumerate`, `sum`.
- 4. Функции определения свойств: `hash`, `id`, `callable`, `issubclass`, `isinstance`, `type`.
- 5. Функции для доступа к внутренним структурам: `locals`, `globals`, `vars`, `intern`, `dir`.
- 6. Функции компиляции и исполнения: `eval`, `execfile`, `reload`, `__import__`, `compile`.
- 7. Функции ввода-вывода: `input`, `raw_input`, `open`.
- 8. Функции для работы с атрибутами: `getattr`, `setattr`, `delattr`, `hasattr`.
- 9. Функции-"украшатели" методов классов: `staticmethod`, `classmethod`, `property`.
- 10. Прочие функции: `buffer`, `slice`.

Группы модулей стандартной библиотеки

- 1. Сервисы периода выполнения. Модули: `sys`, `atexit`, `copy`, `traceback`, `math`, `cmath`, `random`, `time`, `calendar`, `datetime`, `sets`, `array`, `struct`, `itertools`, `locale`, `gettext`.
- 2. Поддержка цикла разработки. Модули: `pdb`, `hotshot`, `profile`, `unittest`, `pydoc`. Пакеты `docutils`, `distutils`.
- 3. Взаимодействие с ОС (файлы, процессы). Модули: `os`, `os.path`, `getopt`, `glob`, `popen2`, `shutil`, `select`, `signal`, `stat`, `tempfile`.
- 4. Обработка текстов. Модули: `string`, `re`, `StringIO`, `codecs`, `difflib`, `mmap`, `sgmllib`, `htmllib`, `htmlentitydefs`. Пакет `xml`.
- 5. Многопоточные вычисления. Модули: `threading`, `thread`, `Queue`.
- 6. Хранение данных. Архивация. Модули: `pickle`, `shelve`, `anydbm`, `gdbm`, `gzip`, `zlib`, `zipfile`, `bz2`, `csv`, `tarfile`.
- 7. Платформо-зависимые модули. Для UNIX: `commands`, `pwd`, `grp`, `fcntl`, `resource`, `termios`, `readline`, `rlcompleter`. Для Windows: `msvcrt`, `_winreg`, `winsound`.
- 8. Поддержка сети. Протоколы Интернет. Модули: `cgi`, `Cookie`, `urllib`, `urlparse`, `httplib`, `smtplib`, `poplib`, `telnetlib`, `socket`, `asyncore`. Примеры серверов: `SocketServer`, `BaseHTTPServer`, `xmlrpclib`, `asynchat`.
- 9. Поддержка Internet. Форматы данных. Модули: `quopri`, `uu`, `base64`, `binhex`, `binascii`, `rfc822`, `mimetools`, `MimeWriter`, `multifile`, `mailbox`. Пакет `email`.
- 10. Python о себе. Модули: `parser`, `symbol`, `token`, `keyword`, `inspect`, `tokenize`, `pyclbr`, `py_compile`, `compileall`, `dis`, `compiler`.
- 11. Графический интерфейс. Модуль `Tkinter`.

Модули math и cmath

- Математические функции для действительных и комплексных аргументов. Это те же функции, что используются в языке Си! Ниже даны функции модуля math. Там, где аргумент обозначен буквой z, аналогичная функция определена и в модуле cmath.
  - `acos(z)` арккосинус `z`
  - `asin(z)` арксинус `z`
  - `atan(z)` арктангенс `z`
  - `atan2(y,x)` значение `atan(y/x)` в радианах
  - `ceil(x)` наименьшее целое, большее или равное
  - `cos(z)` косинус `z`
  - `cosh(x)` гиперболический косинус `x`
  - `e` константа `e`
  - `exp(z)` экспонента (`e**z`)
  - `fabs(x)` абсолютное значение `x`
  - `floor(x)` наибольшее целое, меньшее или равное
  - `fmod(x,y)` остаток от деления `x` на `y`

Модули math и cmath

- `frexp(x)` возвращает мантиссу и порядок `x` как пару `(m, i)`, где `m` - число с плавающей точкой, а `i` - целое, такое, что `x = m * 2.**i`.
- `hypot(x,y)` возвращает `sqrt(x*x + y*y)`
- `ldexp(m,i)` возвращает `m * (2**i)`
- `log(z)` натуральный логарифм `z`
- `log10(z)` десятичный логарифм `z`
- `modf(x)` возвращает пару `(y,q)` - целую и дробную часть `x`.
- `pi` константа пи
- `pow(x,y)` возвращает `x**y`
- `sin(z)` синус `z`
- `sinh(z)` гиперболический синус `z`
- `sqrt(z)` корень квадратный от `z`
- `tan(z)` тангенс `z`
- `tanh(z)` гиперболический тангенс `z`

Модуль random

- Модуль генерирует псевдослучайные числа для нескольких различных распределений.
- Наиболее используемые функции:
  - `random()` – генерирует псевдослучайное число.
  - `choice(s)` – выбирает случайный элемент из последовательности `s`.
  - `shuffle(s)` – размешивает элементы изменчивой последовательности `s` на месте.
  - `randrange([start,] stop[, step])` – выдает случайное целое число из диапазона `range(start, stop, step)`. Аналогично `choice(range(start, stop, step))`.
  - `normalvariate(mu, sigma)` – выдает число из последовательности нормально распределенных псевдослучайных чисел. Здесь `mu` - среднее, `sigma` - среднеквадратическое отклонение `(sigma > 0)`.

Модули операционной системы.

- os Интерфейсы операционной системы
- io Основные средства для работы с потоками
- time Работа со временем и преобразования
- optparse Парсер параметров командной строки
- logging Средства ведения журнала
- getpass Портируемые средства ввода пароля
- curses Поддержка терминала для символьного вывода
- platform Работа с идентификационными данными платформы
- ctypes Библиотека для работы с внешними функциями
- select Ожидание завершения ввода/вывода
- threading Высокоуровневый интерфейс программных потоков
- subprocess Управление подпроцессами
- multiprocessing Интерфейс программных потоков уровня процессов

Модуль os. Назначение. Константы.

- Одна из самых важных библиотек Python!
- Предназначена для взаимодействия с операционной системой.
- Работа с файлами, получение информации об интерфейсах операционной системы и другое.
- Разделители каталогов и другие связанные с этим обозначения доступны в виде констант:
  - `os.curdir` Текущий каталог
  - `os.pardir` Родительский каталог
  - `os.sep` Разделитель элементов пути
  - `os.altsep` Другой разделитель элементов пути
  - `os.pathsep` Разделитель путей в списке путей
  - `os.defpath` Список путей по умолчанию
  - `os.linesep` Признак окончания строки

Модуль os. Работа с файлами и каталогами.

- `access(path, flags)` Проверка доступности файла или каталога с именем `path`.
- `chdir(path)` Делает `path` текущим рабочим каталогом.
- `getcwd()` Текущий рабочий каталог.
- `chmod(path, mode)` Устанавливает режим доступа к `path` в значение `mode`. Следует заметить, что `chmod()` не дополняет действующий режим, а устанавливает его заново.
- `listdir(dir)` Возвращает список файлов в каталоге `dir`.
- `mkdir(path[, mode])` Создает каталог `path`.
- `makedirs(path[, mode])` Аналог `mkdir()`, создающий все необходимые каталоги, если они не существуют. Возбуждает исключение, когда последний каталог уже существует.
- `remove(path)`, `unlink(path)` Удаляет файл `path`. Для удаления каталогов используются `rmdir()` и `removedirs()`.

Модуль os. Работа с файлами и каталогами.

- `rmdir(path)` Удаляет пустой каталог `path`.
- `removedirs(path)` Удаляет `path` до первого непустого каталога. В случае если самый последний вложенный подкаталог в указанном пути - не пустой, возбуждается исключение `OSError`.
- `rename(src, dst)` Переименовывает файл или каталог `src` в `dst`.
- `renames(src, dst)` Аналог `rename()`, создающий все необходимые каталоги для пути `dst` и удаляющий пустые каталоги пути `src`.
- `stat(path)` Возвращает информацию о `path` в виде не менее чем десятиэлементного кортежа. Для доступа к элементам кортежа можно использовать константы из модуля `stat`, например `stat.ST_MTIME` (время последней модификации файла).
- `utime(path, times)` Устанавливает значения времен последней модификации (`mtime`) и доступа к файлу (`atime`).

Модуль os. Работа с процессами.

- Для работы с процессами модуль os предлагает следующие функции (приведены только некоторые, доступные как в Unix, так и в Windows):
  - `abort()` Вызывает для текущего процесса сигнал `SIGABRT`.
  - `system(cmd)` Выполняет командную строку `cmd` в отдельной оболочке, аналогично вызову `system` библиотеки языка C. Возвращаемое значение зависит от используемой платформы.
  - `times()` Возвращает кортеж из пяти элементов, содержащий время в секундах работы процесса, ОС (по обслуживанию процесса), дочерних процессов, ОС для дочерних процессов, а также время от фиксированного момента в прошлом (например, от момента запуска системы).
  - `getloadavg()` Возвращает кортеж из трех значений, соответствующих занятости процессора за последние 1, 5 и 15 минут.

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru