Лекция 3

Введение в Python

Почему Python?

- Низкий порог вхождения
- Качество кода
- Выше скорость разработки
- Высокая переносимость
- Стандартная библиотека

Философия Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea - let's do more of those!

Комментарии

- # - однострочный комментарий
- для многострочных комментариев можно использовать многострочные строковые литералы:

```
"""многострочный
строковый
литерал"""
```

Некоторые базовые типы данных

- Числа
- Строки
- Списки
- Кортежи
- Словари
- Множества
- None
- Логические значения

Функции. Основные сведения

- Функция это подпрограмма, к которой можно обратиться из другого места программы.

Плюсы:

- Многократное использование кода ⇒ отсутствие избыточности
- Декомпозиция программы

Определение функции

- def <имя_функции>(<аргументы>):

  <инструкции>

- return
  - можно возвращать несколько переменных
- pass или ...

Варианты определения функции

- def func($name_1$, $name_2$, ... $name_n$):
- Определение функции с значениями по умолчанию:

  def func($name_1$=val_1, $name_2$=val_2, ... $name_n$=val_n):
- Сначала должны следовать аргументы функции **без** значения по умолчанию.

Вызов функции

- Вызов функции с позиционными аргументами:

  $func(val_1, val_2, \dots val_n)$

- Вызов функции с именованными аргументами:

  $func(name_1=val_1, name_2=val_2, \dots name_n=val_n)$

- Сначала должны следовать **позиционные** аргументы.

Ввод данных

- input()

Функция может принимать на вход текст-сообщение для пользователя, например:

```
input("Введите текст")
```

Функция возвращает **строку.**

Вывод данных

- print([arg$_1$, ...][, sep=' '][, end='\n'][, file=sys.stdout])
- [arg$_1$, ...] # позиционные аргументы
- [, sep=' '] # именованный аргумент
- [, end='\n']# именованный аргумент
- [, file=sys.stdout] # именованный аргумент
- Функция возвращает **None.**

DEMO

Введение в ООП. Объект

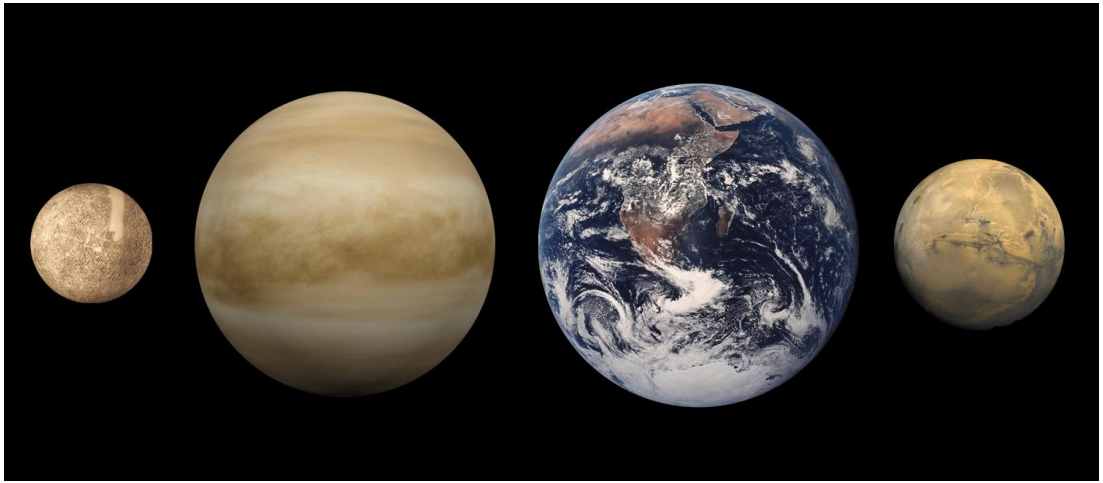
Объект конкретная сущность предметной области.

Введение в ООП. Класс

Класс это **тип** объекта. Или говорят, что объект экземпляр класса.

Класс: **Планета**.

Объекты: **Меркурий, Венера, Земля, Марс**.



Введение в ООП. Метод

Метод - функция класса.

Например: класс **Планета** содержит метод

`получитьСреднийРадиус()`

- Для объекта **Венера** метод вернет 6051,8 км
- Для объекта **Земля** метод вернет 6371,0 км
- Для объекта **Марс** метод вернет 3389,5 км

Пример вызова:

```
Венера.получитьСреднийРадиус()
```

Логические значения

- Логический тип bool
- Значения True, False
- True == 1
- False == 0

Немного о числах

int	Целые числа неограниченной точности
float	Числа с плавающей точкой
complex	Комплексные числа

DEMO

Немного о числах

Представление	Описание
111111111111111	Целое число
2.1e-1, 2E3, 0.123	Вещественное число
0o157, 0x9f, 0b10111	Восьмеричное, шестнадцатеричное и двоичное число
3+4.8j, 22j	Комплексное число

DEMO

Операции
над целыми и вещественными числами

Оператор	Описание
+, -	Сложение, Вычитание
*, /, //, %	Умножение, Деление, Деление с округлением вниз, Остаток от деления
x ** y	Возведение в степень: x^y

Логические операторы

Оператор	Описание
or	Логическое ИЛИ
and	Логическое И
not	Логическое отрицание

Операторы сравнения

Оператор	Описание
x < y, x <= y, x > y, x >= y	Операторы сравнения
x == y, x != y	Операторы проверки на равенство

Модули стандартной библиотеки

- Модуль - файл с программой.
- Модуль может импортировать другие модули для доступа к функциям и переменным.
- Импорт всего модуля:

  `import <имя модуля>`
- Импорт некоторых имен из модуля:

  `from <имя_модуля> импорт <имена>`

DEMO

Строки

- Класс `str`
- Неизменяемый объект
- Для инициализации можно использовать одинарные и двойные кавычки: ‘qwerty’ == “qwerty”

Доступ по индексу

- my_str = ‘AQBWCFD’
- my_str[index] # доступ по индексу

A	Q	B	W	C	F	D
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

Извлечение срезов

- my_str = ‘AQBWCFD’
- my_str[i:j:k] - извлечение среза из my_str. Извлекаются символы от **i** до **j-1** с шагом **k**.
- По умолчанию i = 0, j - длина строки, k = 1.
- Пример: my_str[2:4] = “BW“

A	Q	B	W	C	F	D
0	1	2	3	4	5	6

Извлечение срезов

- my_str = ‘AQBWCFD’
- При **k<0** порядок использования i и j должен быть изменен на противоположный: mystr[j:i:k]
- Пример: my_str[::-1] извлечет все элементы my_str в обратном порядке: j - длина строки, i = 0

A	Q	B	W	C	F	D
-7	-6	-5	-4	-3	-2	-1

Операторы сравнения

Оператор	Описание
in, not in	Проверка вхождения
x < y, x <= y, x > y, x >= y	Операторы сравнения
x == y, x != y	Операторы проверки на равенство

DEMO

Списки

- Список набор упорядоченных разнородных объектов
  - Пример: example_list = [‘hello’, 1, 2, [‘world’, ‘!’]]
- Изменяемый объект
- Произвольное число уровней вложенности:

matrix = [

[0, 1, [2], 3, 4], [0, [1, 2], 3, 4],

[0, 1, [2, 3], 4], [0, 1, 2, 3, 4]]

DEMO

Приведение типов

int()	Приведение к целому числу
float()	Приведение к числу с плавающей точкой
bool()	Приведение к объекту логического типа
str()	Приведение к строке

Оператор ветвления

```
if <выражение_1>:
    <инструкции>
elif <выражение_2>:
    <инструкции>
else:
    <инструкции>
```

Базовые конструкции языка

- break - прекращает выполнение цикла
- continue - прекращает выполнение итерации цикла

Цикл while

```
while <тест1>:
    <инструкции>
    if <тест2>: break # необязательная часть
    if <тест3>: continue # необязательная часть
else: # необязательная часть
    <инструкции>
```

Цикл for

```
for <переменная> in <объект>:
    <инструкции>
    if <тест2>: break # необязательная часть
    if <тест3>: continue # необязательная часть
else:  # необязательная часть
    <инструкции>
```

DEMO

Источники

- https://docs.python.org/3/reference/expressions.html#operator-precedence Таблица приоритетов операторов