

# Python 3. Лекция 1.

## Введение. Основные структуры языка

Емельянов Антон

<https://github.com/king-menin/python-course>

МИРТ

12 февраля 2020 г.

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

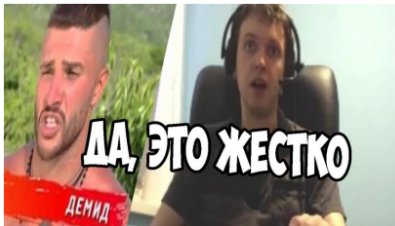
Функции

# План курса

- ▶ 11 лекций
- ▶ N домашних заданий (*HW*)
- ▶ Оценка выставляется следующим образом:

$$mark = \sum_i^N mark(HW_i) + ExtraPoints$$

- ▶ Deadlines:
  - ▶ жОский: 2 недели на выполнение 1 дз.
  - ▶ мягкий: далее максимальный балл за дз умножается на 0.5 (сдавать дз можно до конца курса, но с пониженными баллами).



# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

Функции

# О языке

- ▶ Создан в 1991 году Гвидо ван Россумом
- ▶ Простота использования
- ▶ Свободный и имеет открытый исходный код
- ▶ Высокоуровневый
- ▶ Динамическая типизация

## ► Интерпретируемый

```
(base) ubuntu@vck-u024-vm010:~/python-course/lecture 1. intro-python$ python3 -m dis hello.py
1      0 LOAD_NAME           0 (print)
      2 LOAD_CONST          0 ('hello')
      4 CALL_FUNCTION        1
      6 POP_TOP
      8 LOAD_CONST          1 (None)
     10 RETURN_VALUE
```

## ► Объектно-ориентированный – все является объектом

```
>>> def say(phrase):  
...     print(phrase)  
>>> say("hello")  
hello  
>>> type(say)  
<class 'function'>  
>>> dir(say)  
['_annotations_', '_call_', '_class_', '_closure_', '_code_', '_defaults_', '_delattr_', '_dict_', '_dir_  
r_', '_doc_', '_eq_', '_format_', '_ge_', '_get_', '_getattr_', '_globals_', '_gt_', '_hash_', '_di  
_init_', '_init_subclass_', '_kwdefaults_', '_le_', '_lt_', '_module_', '_name_', '_ne_', '_new_', '_  
qualname_', '_reduce_', '_reduce_ex_', '_repr_', '_setattr_', '_sizeof_', '_str_', '_subclasshook_']
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

Функции



# Основные типы данных и операции

- ▶ Типы: int (long<sup>1</sup>), float
- ▶ Арифметические операции:

```
+ - \ * * * % \ \  
In [5]: 10 / 3 * (5 % 3) // 3 + 1  
Out[5]: 3.0
```

- ▶ Бинарные операции:
  - ▶ & битовое И (AND),
  - ▶ | битовое ИЛИ (OR),
  - ▶ ^ битовое ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR),
  - ▶ ~ битовое ОТРИЦАНИЕ (NOT) — унарная операция,
  - ▶ <<, >> — битовые сдвиги.

```
In [5]: 10 / 3 * (5 % 3) // 3 + 1  
Out[5]: 3.0
```

---

<sup>1</sup>All integers are implemented as “long” integer objects of arbitrary size.

# Логические операции

- ▶ `<`            `<=`            `==`
- ▶ `>`            `>=`            `!=`
- ▶ `not`            `and`            `or`

```
In [16]: 25 < 7 and 0  
Out[16]: False
```

```
In [17]: 25 > 7 or 4  
Out[17]: True
```

```
In [21]: 10 or False  
Out[21]: 10
```

```
In [22]: not False  
Out[22]: True
```

# Тип str

```
In [67]: str.  
str.capitalize str.endswith str.index str.isidentifier str.istitle str.lstrip str.rindex str.split str.title  
str.casefold str.expandtabs str.isalnum str.islower str.isupper str.maketrans str.rjust str.splitlines str.translate  
str.center str.find str.isalpha str.isnumeric str.join str.partition str.rpartition str.startswith str.upper  
str.count str.format str.isdecimal str.isprintable str.ljust str.replace str.rsplit str.strip str.zfill  
str.encode str.format_map str.isdigit str.isspace str.lower str.rfind str.rstrip str.swapcase
```

```
In [70]: "{} win {}".format("Jack", "1000$")  
Out[70]: 'Jack win 1000$'
```

```
In [71]: test_s = "hello world"
```

```
In [72]: test_s[:5]  
Out[72]: 'hello'
```

```
In [73]: test_s[-1]  
Out[73]: 'd'
```

```
In [74]: len(test_s)  
Out[74]: 11
```

```
In [75]: test_s[:-1]  
Out[75]: 'hello worl'
```

## Немного про последовательность выполнения операций

```
In [216]: False == False in [False]  
Out[216]: True
```

```
In [217]: False == (False in [False])  
Out[217]: False
```

```
In [218]: (False == False) in [False]  
Out[218]: False
```

## Немного про последовательность выполнения операций

```
In [216]: False == False in [False]  
Out[216]: True
```

```
In [217]: False == (False in [False])  
Out[217]: False
```

```
In [218]: (False == False) in [False]  
Out[218]: False
```

```
In [219]: x = 5
```

```
In [220]: 1 < x < 10  
Out[220]: True
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

Функции

# Условные конструкции if

- ▶ Легко создавать условные конструкции

```
In [66]: if a > 5:
...:     print(">5")
...: elif a > 10:
...:     print(">10")
...: else:
...:     print("WHY?!")
>5
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

Функции



# Циклы While и for

```
In [49]: a = 5
```

```
In [50]: while a > 0:
...:     a -= 2
...:     print(a)
...: else:
...:     print("Done")
...:
```

```
3
1
-1
Done
```

```
In [60]: n = 99910
```

```
In [61]: length = 0
...: while True:
...:     length += 1
...:     n //= 10
...:     if n == 0:
...:         break
...: print('Длина числа равна', length)
...:
```

Длина числа равна 5

```
In [56]: a = 10
```

```
In [57]: b = 5
```

```
In [58]: while a > 0:
...:     a -= 3
...:     if b > a:
...:         print("b > a", b, a)
...:         break
...:     b -= 1
...: else:
...:     print("Done")
```

```
b > a 3 1
```

```
In [130]: for el in range(5):
...:     print("{}^2={}".format(el, el**2))
```

```
0^2=0
1^2=1
2^2=4
3^2=9
4^2=16
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

**Переменные и объекты**

Контейнеры

Функции

# Переменные

- ▶ Любой объект является ссылкой
- ▶ Типом объекта является то, на что он ссылается
- ▶ Тип объекта может произвольно меняться по ходу выполнения кода, когда ссылка начинает ссылаться на другой объект (например, в результате операции присвоения).

```
In [42]: a = 10
```

```
In [43]: id(a)  
Out[43]: 1793440496
```

```
In [44]: a = 10
```

```
In [45]: id(a)  
Out[45]: 1793440496
```

```
In [46]: a = 10/3
```

```
In [32]: x = 24
```

```
In [33]: id(x)  
Out[33]: 1793440944
```

```
In [34]: x = "test"
```

```
In [35]: id(x)  
Out[35]: 420475560824
```

```
In [36]: y = 5
```

```
In [37]: z = 6
```

```
In [38]: type(y)  
Out[38]: int
```

```
In [39]: y = 5/6
```

```
In [40]: y  
Out[40]: 0.8333333333333334
```

```
In [41]: type(y)  
Out[41]: float
```

# Свойства объектов

- ▶ Любой объект является ссылкой
  - ▶ Identity (`id()` и `is`) не изменяются
  - ▶ Type (`type()` и `isinstance`) не изменяется
  - ▶ Value – по-разному
- ▶ Заботиться о времени жизни объектов самостоятельно не стоит, с этим справляется встроенный в интерпретатор Garbage Collector.

# Изменяемые и неизменяемые объекты

- ▶ Из-за ссылочной структуры многие неизменяемые
  - ▶ Неизменяемые — int, float, complex, bool, str, tuple, frozenset
  - ▶ Изменяемые — list, dict, set
- ▶ При попытке совершить мутирующую операцию с неизменяемым объектом может произойти одна из двух вещей:
  - ▶ Произойдет создание измененной копии объекта (например += )
  - ▶ Произойдет ошибка (оператор [])

```
In [77]: a = 100
In [78]: id(a)
Out[78]: 1793443376
In [79]: a = 10/3
In [80]: id(a)
Out[80]: 420512099544
```

```
In [98]: id1 = id(a)
In [99]: a = 10
In [100]: id2 = id(a)
In [101]: id1==id2
Out[101]: False
In [102]: a += 1
In [103]: id2==id(a)
Out[103]: False
```

```
In [91]: a = frozenset({1, "2"})
In [92]: len(a)
Out[92]: 2
In [93]: a.add(1)
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-93-1742cbb36ce0> in <module>()
----> 1 a.add(1)
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

**Контейнеры**

Функции

# Контейнеры

- ▶ При создании двух mutable-объектов отдельно - они будут гарантированно разными. Для immutable объектов это верно не всегда.
  - ▶ Контейнер - это объект, содержащий ссылки на другие объекты.
- 
- ▶ list – динамический массив
  - ▶ tuple – кортеж
  - ▶ dict – словарь (хеш-мэп)
  - ▶ set – множество

```
In [104]: list1 = []  
In [105]: list2 = []  
  
In [106]: list1 == list2  
Out[106]: True  
  
In [107]: list1 is list2  
Out[107]: False  
  
In [108]: a = 1  
In [109]: b = 1  
  
In [110]: id(a) == id(b)  
Out[110]: True  
  
In [111]: a is b  
Out[111]: True
```

# Контейнеры

```
In [141]:
...: lst = [1, 123, [1, 2, 3], 4]
...:
...: lst = [] # list()
...:
...: lst.append(5)
...:
...: lst += [10, 11]
...:
...: lst[0] = "changed"
...:
...: lst
...:
Out[141]: ['changed', 10, 11]

In [142]: lst.remove(10)

In [143]: del lst[0]

In [144]: lst
Out[144]: [11]

In [145]: lst.insert(0, "test")

In [146]: lst
Out[146]: ['test', 11]
```

---

```
In [175]: a = dict(one=1, two=2)
```

```
In [176]: a
Out[176]: {'one': 1, 'two': 2}
```

```
In [177]: {1: 'a', 2: 'b'}
Out[177]: {1: 'a', 2: 'b'}
```

```
In [149]: tpl = tuple() # immutable
```

```
In [150]: date = ('2017', '9', '20')
```

```
In [151]: tpl = tuple([1, 2, 3])
```

```
In [152]: tpl
Out[152]: (1, 2, 3)
```

---

```
In [163]: st = {1, 2, 3}
```

```
In [164]: st = set(range(5))
```

```
In [165]: st
Out[165]: {0, 1, 2, 3, 4}
```

```
In [166]: st.update({"test", 5})
```

```
In [167]: st.add(2)
```

```
In [168]: st.add(6)
```

```
In [169]: st
Out[169]: {0, 1, 2, 3, 4, 'test', 5, 6}
```



# Индексирование и срезы

- ▶ Работает со списками, кортежами и строками, при некотором старании и с пользовательскими типами.

```
In [194]: lst = list(range(20))
```

```
In [195]: lst[1]
```

```
Out[195]: 1
```

```
In [196]: lst[-1]
```

```
Out[196]: 19
```

```
In [197]: lst[-4]
```

```
Out[197]: 16
```

```
In [198]: lst[1:5]
```

```
Out[198]: [1, 2, 3, 4]
```

```
In [199]: lst[1:14:2]
```

```
Out[199]: [1, 3, 5, 7, 9, 11, 13]
```

```
In [200]: lst[-1:1:-2]
```

```
Out[200]: [19, 17, 15, 13, 11, 9, 7, 5, 3]
```

```
In [201]: lst[::-2]
```

```
Out[201]: [19, 17, 15, 13, 11, 9, 7, 5, 3, 1]
```

```
In [202]: lst[:]
```

```
Out[202]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

## Распаковка и for

```
In [207]: for a, b in [(1, 3), (2, 4)]:  
...:     print(a, b)
```

```
1 3  
2 4
```

```
In [208]: lst = "abcd"
```

```
In [210]: for i, sym in enumerate(lst):  
...:     print(i, sym)
```

```
0 a  
1 b  
2 c  
3 d
```

```
In [213]: lst2 = list(range(4))
```

```
In [214]: for el1, el2 in zip(lst, lst2):  
...:     print(el1, el2)
```

```
a 0  
b 1  
c 2  
d 3
```

## Генераторы стандартных контейнеров

```
[i**2 for i in range(100) if not i % 3] # list
```

```
[i * j for i in range(5) for j in range(5)]
```

```
{i for i in range(10) if i > 3} # set
```

```
{i : i ** 3 for i in range(10)} # dict
```

## Полезные контейнеры

```
In [228]: from collections import defaultdict, Counter
```

```
In [229]: a = Counter()
```

```
In [230]: a['test'] += 1
```

```
In [231]: b = defaultdict(str)
```

```
In [232]: b['test'] += 'hi'
```

```
In [233]: b
```

```
Out[233]: defaultdict(str, {'test': 'hi'})
```

# План лекции

Об этом курсе

Об языке

Основные типы данных и операции

Условные конструкции if

Циклы While и for

Переменные и объекты

Контейнеры

Функции

## Функции

```
In [242]: def print_args(*args, **kwargs):  
...:     print("args:")  
...:     for arg in args:  
...:         print(arg)  
...:     print("\nkwargs:")  
...:     for key, val in kwargs.items():  
...:         print(key, val)
```

```
In [243]: print_args(1, 2, 3, one=1, two=2)
```

args:

1

2

3

kwargs:

one 1

two 2

# Домашнее задание 1

- ▶ Целью этого задания является знакомство со стандартными контейнерами и функциями
- ▶ Deadline (получение полных баллов): 27.02.2020
- ▶ Адрес: login-const@mail.ru
- ▶ Реализовать функцию, которая имеет один позициональный (texts) и один необязательный keyword аргумент (dictionary). Вернуть функция должна обновленный словарь слов dictionary, где каждому слову соответствует его уникальный индекс. При этом возвращается в функции всегда второй аргумент. Если передан второй аргумент, то он меняется (см. пример)

```
In [121]: 1 d1 = create_dict(["Что, как"])
          2 d1
```

```
Out[121]: {'Что,': 0, 'как': 1}
```

```
In [122]: 1 d2 = create_dict(["Что , как", "нормально а ти Что"], d1)
          2 d2, d1 is d2
```

```
Out[122]: ({'Что,': 0, 'как': 1, 'Что': 2, ',': 3, 'нормально': 4, 'а': 5, 'ти': 6}, True)
```

Спасибо за внимание