

Интенсив Python

Лекция 2

Подробнее про типы, управляющие конструкции

Кандауров Геннадий



образование

Напоминание отметиться на портале

+ ОСТАВИТЬ ОТЗЫВ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3
Введение в Python, основные
понятия, тестирование
Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

ПодписатьсяСоздать топик

Поиск по авторам, заголовку и тексту топика...Найти

Добро пожаловать на курс!

Углубленный Python

ИзменитьУдалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад
Углубленный Python → Добро пожаловать
на курс! 0

Екатерина Черкасова 7 дней назад
Стажировка → Приглашаем мобильных,
фронтенд- и бэкэнд-разработчиков на
Weekend Offer! 0

Дарья Вовченко 9 дней назад
Углубленный Python → Добро пожаловать
в образовательные проекты VK
Образование! 0

Дарья Вовченко 9 дней назад
Разработка веб-сервисов на

Содержание занятия

1. Типы данных
2. Управляющие конструкции
3. Интерактив

Переменные

`num = 1` # операция присваивания переменной `num` значения `1`

- допустимые символы: буквы, цифры, `_` (подчеркивание)
- начинаться только с буквы или `_`

valid

`num = 1`

`_num = 1`

`__num = 1`

`число1_2_34 = 1`

`_123_число = 1`

invalid

`1num = 1`

`1_num = 1`

Базовые типы: числа

Целые числа (int)

```
num = 42
```

```
num = 42_000_000
```

```
num = 0o10 # 8
```

```
num = 0x10 # 16
```

Вещественные числа (float)

```
num = 3.14
```

Комплексные числа (complex)

```
num = 9 + 0.75j # num.real, num.imag
```

Арифметические операции

- Сложение: +
- Вычитание: -
- Умножение: *
- Деление: /
- Целочисленное деление: //
- Остаток от деления: %
- Возведение в степень: **

Битовые операции

- Умножение: &
- Сложение: |
- Инверсия: ~
- Исключающее ИЛИ: ^
- Побитовый сдвиг: <<, >>

Базовые типы: bool

Логический тип (bool)

`is_true = True` # или `False`

- Логическое И: **and**
- Логическое ИЛИ: **or**
- Отрицание: **not**

Вычисление составных логических выражений **ленивое**:

`return_false()` and `return_true()`

`return_true()` or `return_false()`

Базовые типы: строки

```
s = "просто строка" # str  
byte_s = b"qwerty" # bytes  
raw_s = r"111\нданные" # str
```

```
str.encode() -> bytes
```

```
bytes.decode() -> str
```

```
str1 + str2 # конкатенация
```

```
"a" * 5 # повторение "aaaaa"
```

Форматирование

- "x %s %d" % ("y", 42)
- "x %(y)s %(n)d" % dict(y="y", n=42)
- "x {} {}".format("y", 42)
- y, n = "y", 42

```
f"x {y} {n}"
```

```
"x y 42"
```

Базовые типы: коллекции

Список (list)

```
lst = []
```

```
lst = [1, 2, 3]
```

```
lst1 = list(lst)
```

```
lst.append(4)
```

Кортеж (tuple)

```
tup = ()
```

```
tup = (1,)
```

```
tup = 1, 2, 3
```

```
tup = tuple(lst)
```

Swap

```
a, b = b, a
```

Распаковка

```
lst = [1, 2, 3, 4, 5]
```

```
a, b, *c = lst # 1, 2, [3, 4, 5]
```

```
a, *b, c = lst # 1, [2, 3, 4], 5
```

```
lst = ["123", "456"]
```

```
(a, *b), c = lst
```

```
# "1", ["2", "3"], "456"
```


Базовые типы: коллекции

Словарь (dict)

```
d = {} # dict()
d1 = {1: 11}
d1[2] = 22
d2 = dict(x=10, y=20)
d3 = {**d1, **d2, 3: 33}
```

Множество (set, frozenset) уникальных неизменяемых объектов.
Множество не индексируется, но по нему можно итерироваться.

```
s = set()
s = {1, 2, 3}
s.add(4)
s = set("123123123") # {"1", "2", "3"}
```

Базовые типы: коллекции

```
lst = [1, 3, 2]
```

```
dct = {1: 11, 2: 22}
```

- Оператор in:

```
1 in lst
```

```
1 in dct
```

```
"456" in "123456"
```

- Сортировка:

```
sorted(dct)
```

```
lst.sort()
```

- Итерирование:

```
for key in dct: pass
```

Списковые включения (компрехеншены)

```
lst = [n ** 2 for n in range(10) if n % 2]
```

```
s = {ch for ch in "abcabcbca"}
```

```
dct = {n: n ** 2 for n in range(10) if n % 2}
```

```
gen = (n ** 2 for n in range(10) if n % 2)
```

Изменяемые, неизменяемые типы

Неизменяемые

- int, float, bool, complex
- str, bytes
- tuple
- frozenset

Изменяемые

- list
- dict, set
- user defined

Управляющие конструкции: циклы

Цикл `for`

```
for elem in sequence:
    process(elem)
    # break
    # continue
else:
    do_else()
```

Цикл `while`

```
while condition:
    process()
    # break
    # continue
else:
    do_else()
```

Управляющие конструкции: условный оператор

```
if cond1:
    action1()
elif cond2:
    action2()
else:
    action3()
```

Тернарный оператор

```
result = action1() if cond else action2()
```

Pattern matching (3.10)

```
value = 42
match value:
    case 41:
        act_41()
    case 42:
        act_42()
    case other:
        act_other(other)
```

Управляющие конструкции: контекстный менеджер

```
class CtxManager:
    def __init__(self, name):
        self.db = connect_db(name)
    def __enter__(self):
        return self.db
    def __exit__(self, exc_type, exc_val, exc_tb):
        self.db.close()
```

```
# from contextlib import contextmanager
```

```
with CtxManager("db_name") as db:
    do_action(db)
```

Функции, lambda-функции

```
def add(a, b):  
    return a + b
```

```
def do_action(action, *args, **kwargs):  
    print(f"do {action} with {args}, {kwargs}")
```

```
    if callable(action):  
        return action(*args, **kwargs)  
    else:  
        return action
```

```
do_action(add, 1, b=2)
```

```
do_action(lambda x, y: x * y, 5, y=6)
```

Функции: декораторы

Декоратор - это функция, которая принимает функцию и возвращает функцию.

```
def deco(fn):  
    def inner(*args, **kwargs):  
        print("before", fn.__name__)  
        res = fn(*args, **kwargs)  
        print("after", fn.__name__)  
        return res  
    return inner
```

```
@deco
```

```
def add_nums(a, b):  
    return a + b
```


Функции: декораторы с параметрами

```
def deco(add_param):  
    def inner_deco(fn):  
        def inner(*args, **kwargs):  
            return fn(*args, **kwargs) + add_param  
        return inner  
    return inner_deco
```

```
@deco(45)
```

```
def add_nums(a, b):  
    return a + b
```

```
add_nums = deco(add_nums)
```

Итераторы

Итератор представляет собой объект-перечислитель, который для данного объекта выдает следующий элемент, либо вызывает исключение, если элементов больше нет.

```
num_list = [1, 2, 3]
itr = iter(num_list)
print(next(itr))  # 1
print(next(itr))  # 2
print(next(itr))  # 3
print(next(itr))  # StopIteration
```

```
class SpecialIterator:
    def __init__(self, limit):
        self.limit = limit
        self.counter = 0
    def __next__(self):
        if self.counter < self.limit:
            self.counter += 1
            return self.counter
        else:
            raise StopIteration

iter_obj = SpecialIterator(3)
print(next(iter_obj))
```

Функции: генераторы

Генератор – функция, которая при вызове `next()` возвращает следующий объект по алгоритму ее работы. Вместо `return` в генераторе используем `yield` (или вместе).

```
def gen(count):  
    while count > 0:  
        yield count  
        count -= 1  
    return count # будет аргументом StopIteration  
  
for i in gen(5):  
    print(i) # 5, 4, 3, 2, 1
```

Классы

```
class A:
    cls_attr = 42

    def __init__(self, val):
        self.val = val
        self._protected = 10
        self.__private_val = 20

    def print_name(self):
        print(self.__class__.__name__)

    @property
    def private_val(self):
        return self.__private_val
```

```
class B(A):
    cls_attr = 55

    def __init__(self, bval, val):
        super().__init__(val)

        self.bval = val
        self.__private_val = 999

b = B(1, 2)
print(b.private_val) # ???

isinstance(b, (int, A)) # ???
issubclass(B, object) # ???
```

Модули и пакеты

Модули являются основным компонентом организации кода в питоне (и это тоже объекты), объединяются в пакеты.

```
import this
collections = importlib.import_module('collections')
itertools = __import__('itertools')
```

file.py - модуль

--app/ - пакет

|--__init__.py

|--hello.py

Валидные импорты

```
import sys
```

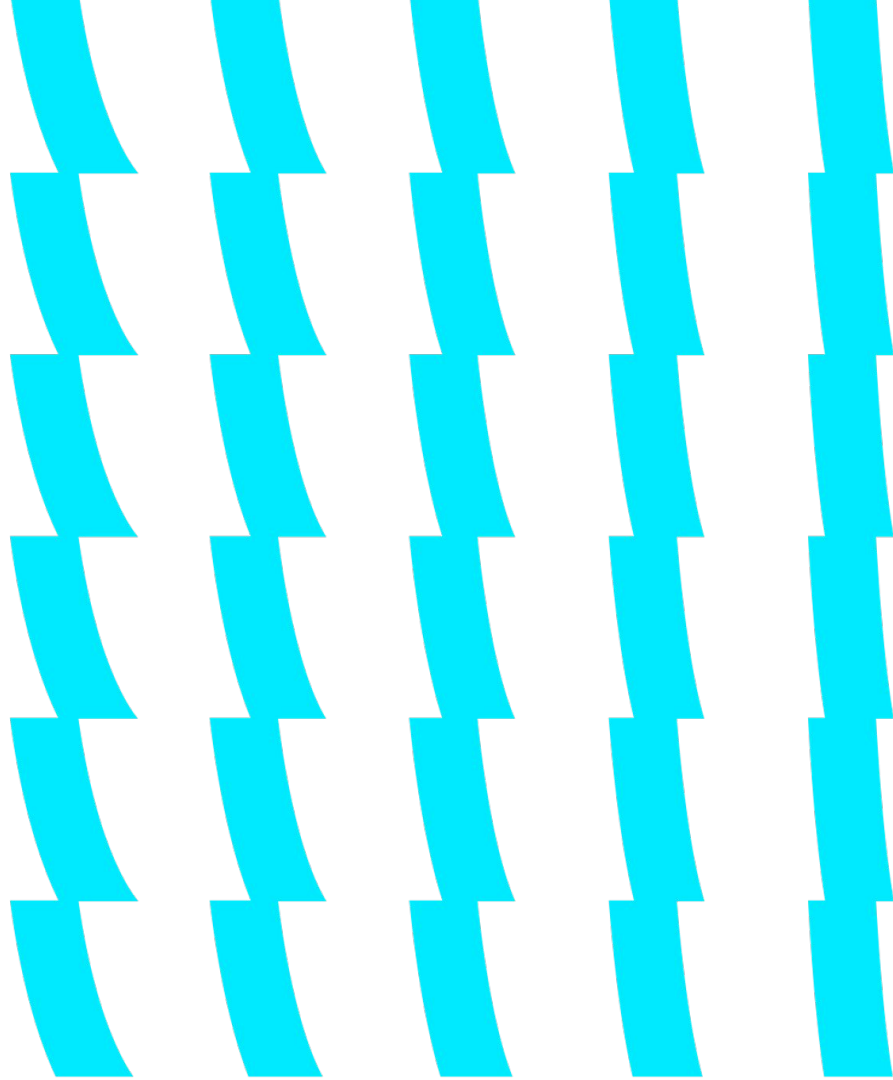
```
from sys import path
```

```
from sys import *
```

```
import sys as s
```

```
from sys import path as sys_path
```

Работа с файлами



Типы операций с файлами

- связанные с его открытием: открытие, закрытие файла, запись, чтение, перемещение по файлу и др.
- выполняющиеся без его открытия: работа с файлом как элементом файловой системы - переименование, копирование, получение атрибутов и др.

Файловый объект

При открытии файла операционная система возвращает специальный дескриптор файла (идентификатор), однозначно определяющий, с каким файлом далее будут выполняться операции.

В Python работа с файлами осуществляется через специальный абстрактный файловый объект. В зависимости от способа создания такого объекта, он может быть привязан как к физическому файлу на диске, так и другому устройству, поддерживающему схожие операции (стандартный ввод/вывод и пр.).

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,  
      newline=None, closefd=True, opener=None)
```

```
# кодировка  
import locale  
locale.getpreferredencoding(False)
```


Файловый объект

```
f = open("some.file", "r")
```

```
data = f.read()
```

```
f.close()
```

```
# лучше
```

```
with open("some.file", "r") as f:
```

```
    data = f.read()
```

Обработка файла

Режим	Описание
r	Только для чтения.
w	Только для записи. Создаст новый файл, если не найдет с указанным именем.
rb	Только для чтения (бинарный).
wb	Только для записи (бинарный). Создаст новый файл, если не найдет с указанным именем.
r+	Для чтения и записи.
rb+	Для чтения и записи (бинарный).
w+	Для чтения и записи. Создаст новый файл для записи, если не найдет с указанным именем.

wb+	Для чтения и записи (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
a	Откроет для добавления нового содержимого. Создаст новый файл для записи, если не найдет с указанным именем.
a+	Откроет для добавления нового содержимого. Создаст новый файл для чтения записи, если не найдет с указанным именем.
ab	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
ab+	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для чтения записи, если не найдет с указанным именем.

Домашнее задание #02

- Определение чисел наиболее близких к нулю
- Слияние последовательностей
- assert-ы

Напоминание отметиться на портале Vol 2

+ ОСТАВИТЬ ОТЗЫВ ПОСЛЕ ЛЕКЦИИ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3
Введение в Python, основные
понятия, тестирование
Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

ПодписатьсяСоздать топик

Поиск по авторам, заголовку и тексту топика...

Найти

Добро пожаловать на курс!

Углубленный Python

ИзменитьУдалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад
Углубленный Python → Добро пожаловать
на курс! 0

Екатерина Черкасова 7 дней назад
Стажировка → Приглашаем мобильных,
фронтенд- и бэкэнд-разработчиков на
Weekend Offer! 0

Дарья Вовченко 9 дней назад
Углубленный Python → Добро пожаловать
в образовательные проекты VK
Образование! 0

Дарья Вовченко 9 дней назад
Разработка веб-сервисов на

Спасибо за
внимание



образование