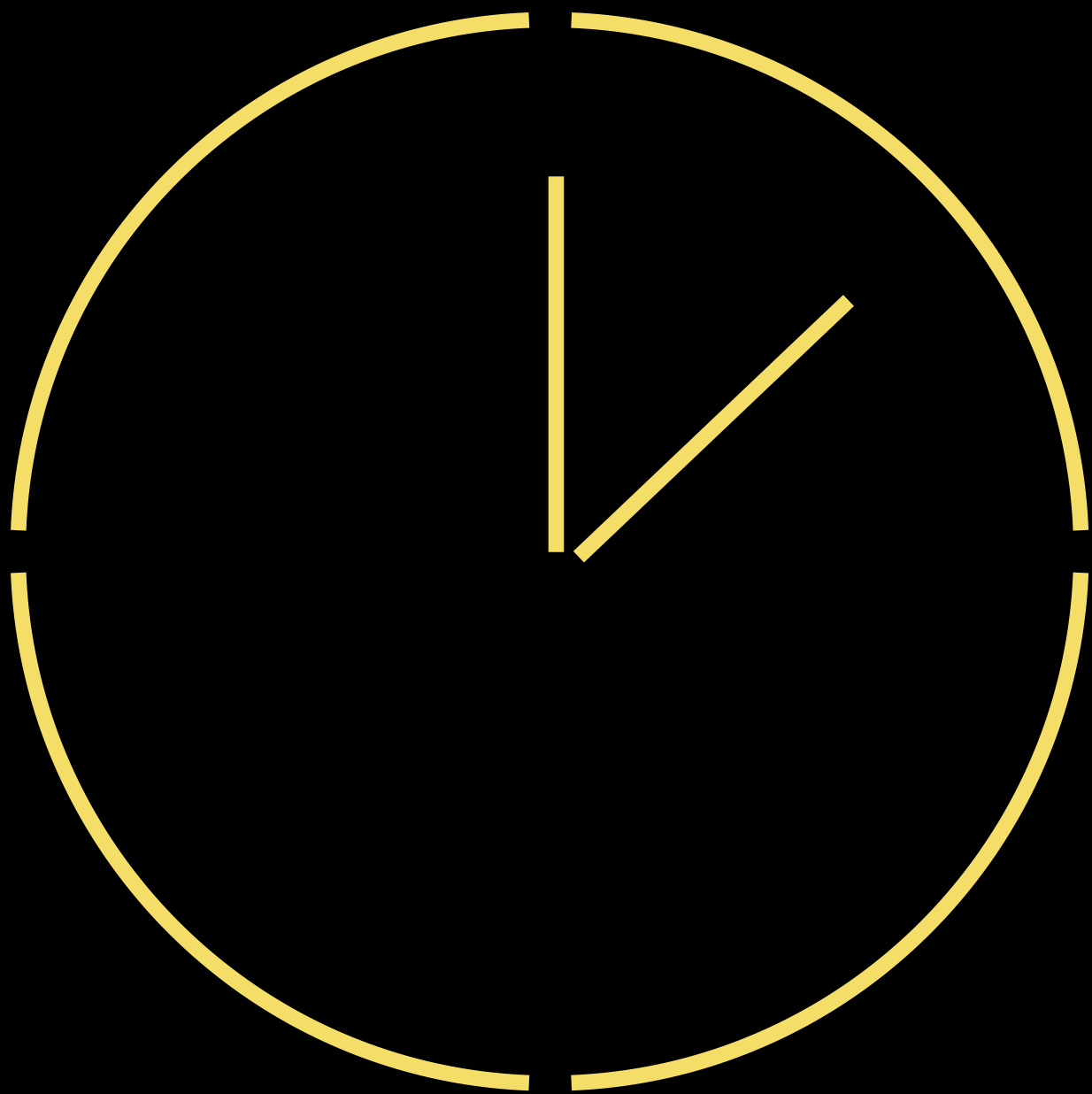


Продвинутый Python

Александр Ольферук,
наставник

Яндекс Практикум

Цели на консультацию



Первая часть – 50 минут

- Знакомство
- Лайфхаки

Перерыв – 10 минут

Вторая часть – 30 минут

- Отвечаю на ваши вопросы по Питону
- Поиграем

Про меня

Computer Vision Engineer

Закончил Воронежский ГУ

Работал с ML, DL, RecSys

Преподавал в ВГУ, проводил
летние школы и факультатив

Катаюсь на лонгборде

Играю в настольные игры

Я – наставник, потому что:

- Хочется помогать, развиваясь самому
- Кажется, у меня получается преподавать :)

Наши договорённости

Организованность

- Будь вовремя
- Понятное имя в Zoom (и везде)
- Камера включена
- Правило одного включенного микрофона
- Вопросы в чате

Комфортная коммуникация

- Общаемся на «ты»
- Вовлеченность и проактивность
- Обучение – это ошибки
- Уважительное отношение
- Критикуешь – предлагай
- Береги общее время
- Запись только для потока

Цель – научиться

- Мы – разные
- Учимся самостоятельно принимать решения
- Самостоятельность – это поиск и общение, а не одиночество
- Взаимопомощь
- Нет спойлерам

Лайфхаки

Комментарии

```
1  def div(a, b):  
2  |      return int(a//b)
```

Комментарии

```
1 def div(a, b):  
2     return int(a//b)
```

```
In [1]: 1 from my_script import div
```

```
In [ ]: 1 div()
```

Signature: div(a, b)

Docstring: <no docstring>

Комментарии

```
1  def div(a, b):  
2  |      return int(a//b)
```

```
def div(a, b):  
    '''  
    Возвращает частное в виде целого (int)  
    :param a: Делимое  
    :param b: Делитель  
    :returns: Частное  
    '''  
    return int(a//b)
```

Больше здесь: <https://www.sphinx-doc.org/en/master/>

Комментарии

```
def div(a, b):  
    '''  
    Возвращает частное в виде целого (int)  
    :param a: Делимое  
    :param b: Делитель  
    :returns: Частное  
    '''  
    return int(a//b)
```

```
In [1]: 1 from my_script import div
```

```
In [ ]: 1 div(|)
```

Signature: div(a, b)

Docstring:

Возвращает частное в виде целого (int)

:param a: Делимое

:param b: Делитель

:returns: Частное

Больше здесь: <https://www.sphinx-doc.org/en/master/>

Quick mafs

```
1    5//2          # 2
2    3.4//1.5      # 2.0
3    int(3.4//1.5) # 2
```

Приоритеты и условные операторы

```
1    **          # сначала это
2    *, /, //    # потом это
3    +, -        # в конце вот это
4
```

Приоритеты и условные операторы

1	**	# сначала это
2	*, /, //	# потом это
3	+, -	# в конце вот это
4		
5	^	# XOR
6	and, or, ^	# в самом конце

Приоритеты и условные операторы

```
1  **          # сначала это
2  *, /, //    # потом это
3  +, -        # в конце вот это
4
5  ^           # XOR
6  and, or, ^   # в самом конце
7  &, |         # НЕ ИСПОЛЬЗУЙТЕ ИХ В if!
8
9  m1 = df['height'] > 100
10 m2 = df['width'] < 50
11 m3 = m1 & m2   # только здесь
12
```

Приоритеты и условные операторы

```
1    **          # сначала это
2    *, /, //    # потом это
3    +, -        # в конце вот это
4
5    ^           # XOR
6    and, or, ^   # в самом конце
7    &, |         # НЕ ИСПОЛЬЗУЙТЕ ИХ В if!
8
9    m1 = df['height'] > 100
10   m2 = df['width'] < 50
11   m3 = m1 & m2   # только здесь
12
13   bool(0)        # False
14   bool(1), bool(-20) # True
```

List comprehension

```
1 [x**2 for x in range(1, 6)] # [1, 4, 9, 16, 25]
```

List comprehension

```
1  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
2
3  [x**2 for x in range(1, 6) if x % 2 == 0]  # [4, 16]
4
```


List comprehension

```
1  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
2
3  [x**2 for x in range(1, 6) if x % 2 == 0]  # [4, 16]
4
5  [(x**2 if x % 2 == 0 else -x) for x in range(1, 6)]  # [-1, 4, -9, 16, -25]
```

List comprehension vs generators

```
1  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
2
3  (x**2 for x in range(1, 6))  # <generator object <genexpr> at 0x7f15b20dca40>
```

Больше здесь: <https://code-maven.com/list-comprehension-vs-generator-expression>

И здесь: <https://stackoverflow.com/questions/47789/generator-expressions-vs-list-comprehension>

List comprehension

```
1  # Получится лист (список)
2  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
3
```

List comprehension

```
1  # Получится лист (список)
2  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
3
4  # Получится множество
5  {x**2 for x in range(1, 6)}  # {1, 4, 9, 16, 25}
6
```

List comprehension

```
1  # Получится лист (список)
2  [x**2 for x in range(1, 6)]  # [1, 4, 9, 16, 25]
3
4  # Получится множество
5  {x**2 for x in range(1, 6)}  # {1, 4, 9, 16, 25}
6
7  # Получится словарь
8  {x: x**2 for x in range(1, 6)}  # {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

List comprehension

```
1  [item for sublist in list for item in sublist]  
2
```

List comprehension

```
1 [item for sublist in list for item in sublist]
2
```

=

```
1 for sublist in list:
2     | for item in sublist:
3     |     | item
```

List comprehension

```
1  [item for sublist in list for item in sublist]  
2  
3  [item for item in sublist for sublist in list]
```


List comprehension

```
1 [item for sublist in list for item in sublist]  
2  
3 [item for item in sublist for sublist in list]
```

Распаковка

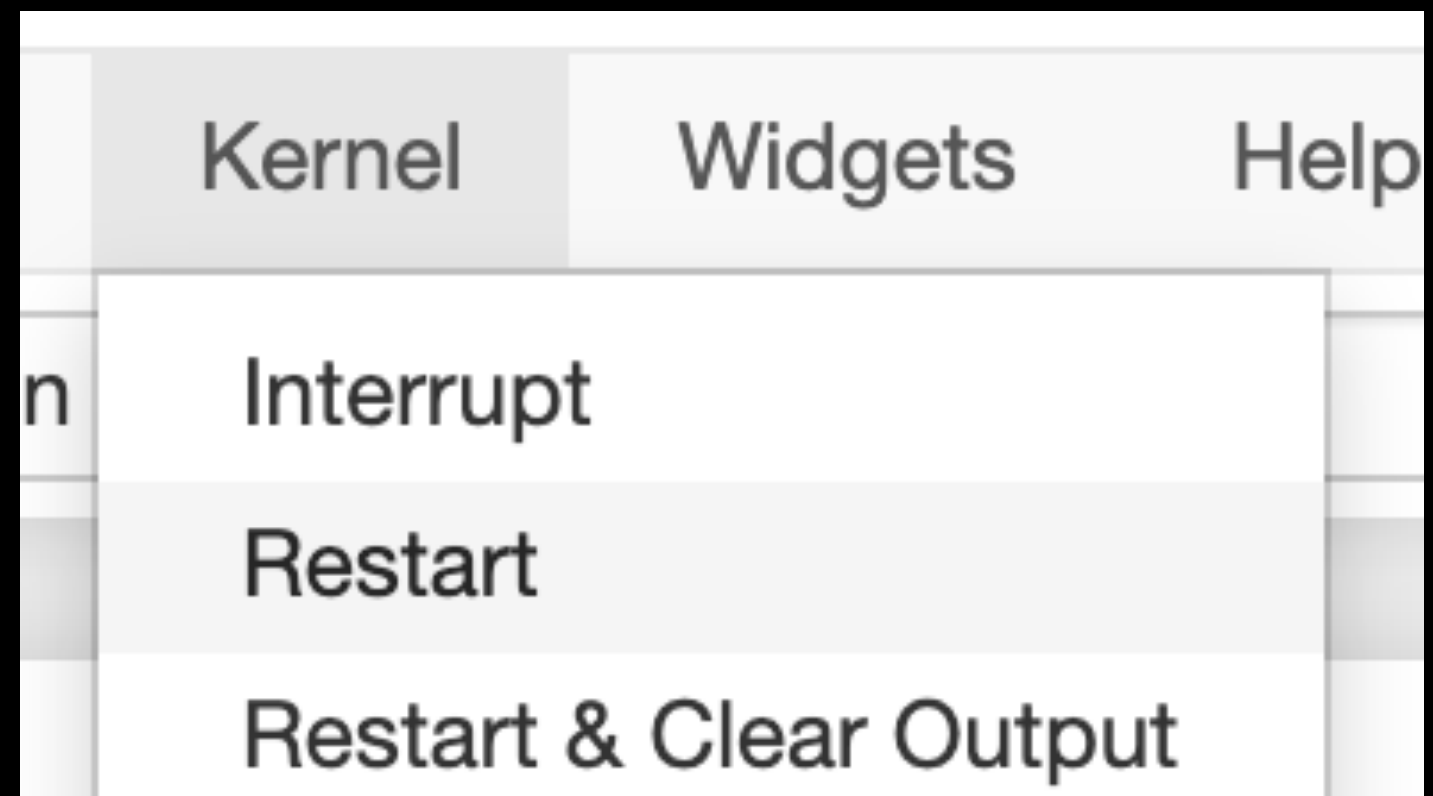
```
1  x = {'a': 1, 'c': 2}
2  y = {'b': 3, 'd': 4}
3  z = {**x, **y}
```

```
1  a = [1, 2]
2  b = [3, 4]
3  c = [5, 6]
4  [*a, *b, *c] # конкатенированный лист
5  (*a, *b, *c) # конкатенированный кортеж
6  {*a, *b, *c} # конкатенированное множество
```

Variables shadowing

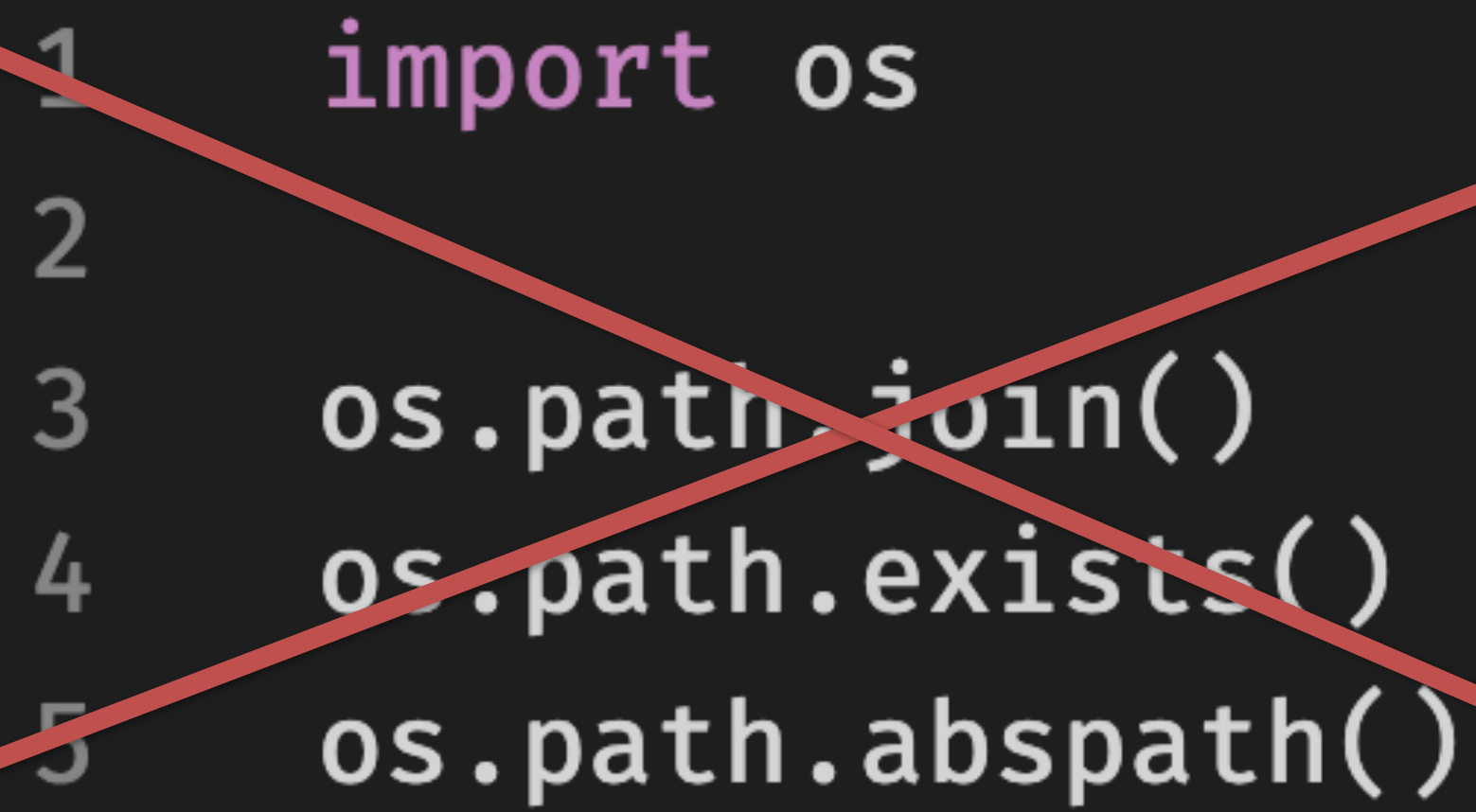
```
1  x = 1
2  # ... many cells ...
3
4  def do_heavy_stuff(q, w):
5      # ... lotsa code ...
6      e = q + w + x
7      # ... lotsa code ...
8      return e
```

Variables shadowing



или просто нажмите на клавиатуре два раза “0”

Pathlib



```
1  import os
2
3  os.path.join()
4  os.path.exists()
5  os.path.abspath()
```

Pathlib

```
1  from pathlib import Path
2
3  p = Path('my-folder')
4  p / 'path' / 'to' / 'other' / 'folder' / 'or' / 'file.txt'
5
```

Pathlib

```
1  from pathlib import Path
2
3  p = Path('my-folder')
4  p / 'path' / 'to' / 'other' / 'folder' / 'or' / 'file.txt'
5
6  p.exists()
7  p.is_dir()
8  p.parts
9  p.with_name('sibling.png') # меняет только имя, сохраняя папку
10 p.with_suffix('.jpg') # меняет только расширение, сохраняя все остальное
11 p.chmod(mode)
12 p.rmdir()
13 sorted(p.glob('*'))
14 sorted(p.glob('**/*.png'))
```

print

```
1  print(a, b, c, sep=' ')\n2  print(a, b, c, sep='\\t')\n3
```


print

```
1  print(a, b, c, sep=' ')\n2  print(a, b, c, sep='\\t')\n3\n4  _print = print # сохраняем оригинальный print на всякий случай\n5  def print(*args, **kwargs):\n6      pass # переопределяем принт: например, пишем в файл или на сервер
```

Для легкого чтения: <https://docs.python.org/3/howto/logging.html>

И вот: <https://docs.python.org/3/library/logging.html>

Strings f-formatting

```
1  f'{np.mean(a):.3f} santimeters in average'  # 150.823
2  f'{i+1:04d}.png'  # 0001.png
```

Подробнее здесь: <https://www.datacamp.com/community/tutorials/f-string-formatting-in-python>

Strings f-formatting

```
f' { "ok" } '  
f" { 'ok' } "  
f" { "not ok" } "  
f' { 'not ok' } '  
f' { {x for x in range(3)} } { "ok" } '
```

```
f' { } { "not ok" } '  
f' \{ } { "not ok" } '
```

Подробнее здесь: <https://www.datacamp.com/community/tutorials/f-string-formatting-in-python>

Проверка на None

```
1  if a is not None:
2      pass
3
4  if a: # неправильная проверка на None!
5      pass
6
7  if a == None: # формально правильная, но нежелательная проверка
8      pass
```

Типизация

```
1  def div(a: int, b: int) → int:  
2  |      return a//b
```

Типизация

```
1  from typing import Union
2
3  any_number = Union[float, int]
4
5  def div(a: any_number, b: any_number) → int:
6      return int(a//b)
```

Статья: <https://docs.python.org/3/library/typing.html>

Типизация

```
1  import enforce
2  from typing import Union
3
4  any_number = Union[float, int]
5
6  @enforce.runtime_validation
7  def div(a: any_number, b: any_number) → int:
8      |    return int(a//b)
```

RuntimeTypeError:

The following runtime type errors were encountered:

Argument 'a' was not of type typing.Union[float, int]. Actual type was str.

Читать про enforce: <https://github.com/RussBaz/enforce>

Функции нужны, когда вы повторяете код

```
goods_items = read_all_lines(goods)
bads_items = read_all_lines(bads)
multiples_items = read_all_lines(multiples)
to_fixs_items = read_all_lines(to_fixs)

goods_items = [Path(x).stem for x in goods_items]
bads_items = [Path(x).stem for x in bads_items]
multiples_items = [Path(x).stem for x in multiples_items]
to_fixs_items = [Path(x).stem for x in to_fixs_items]

df_good = pd.DataFrame(data={'stem': goods_items})
df_bad = pd.DataFrame(data={'stem': bads_items})
df_multiple = pd.DataFrame(data={'stem': multiples_items})
df_to_fix = pd.DataFrame(data={'stem': to_fixs_items})

df_good = pd.merge(df_good, df, how='left')
df_bad = pd.merge(df_bad, df, how='left')
df_multiple = pd.merge(df_multiple, df, how='left')
df_to_fix = pd.merge(df_to_fix, df, how='left')
```


Функции нужны, когда нужно разгрузить код

```
def show(img):  
    plt.figure(figsize=(20, 15))  
    plt.imshow(img)  
    plt.axis('off')  
    plt.show()
```

Перерыв

Жду тебя через 10 минут



The background features several white lines of varying lengths and orientations. One line starts from the top left and extends towards the center. Another line starts from the bottom left and extends towards the center. A third line starts from the bottom left and extends towards the bottom right. A fourth line starts from the top right and extends towards the center. These lines create a sense of dynamic movement and geometric structure.

Поиграем?

Разбор ваших вопросов

Про стайл гайды

<https://flake8.pycqa.org/en/latest/>

<https://www.python.org/dev/peps/pep-0008/>

<https://docs.python-guide.org/writing/style/>

<https://google.github.io/styleguide/pyguide.html>

Домашка

<https://dev.to/devmount/10-awesome-pythonic-one-liners-explained-3doc>

- Читать про try-except-finally
- Ложиться спать до 12 часов ночи
- Читать про itertools
- Читать про toolz
- Делать заминку после бега и любой физухи
- Читать про local, global и scope'ы переменных
- Отказаться от сладкого

далее опционально:

- Читать про замыкания
- Читать про конструкцию yield from
- Читать про enum
- Читать про map, reduce, filter

Продвинутый Python

Александр Ольферук,
наставник

Яндекс Практикум