

# Типизация в Питоне

Как с этим работать?

# Типизация

- Первые упоминания о подсказках типов в языке программирования Python появились в базе Python Enhancement Proposals (PEP-483). Такие подсказки нужны для улучшения статического анализа кода и автодополнения редакторами, что помогает снизить риски появления багов в коде.

# Типизация

- Для обозначения базовых типов переменных используются сами типы:

1. Str
2. Int
3. Float
4. Bool
5. Complex
6. Bytes
7. etc

# Типизация

- Пример использования базовых типов в python-функции:

```
1 def func(a: int, b: float) -> str:  
2     a: str = f"{a}, {b}"  
3     return a  
4     # return a * b  
5  
6     example = func(5, 4.5)  
7     print(example)
```

# Типизация

- Ну либо такой пример:

```
main.py x
1  def func(a: int, b: float):
2      # a: str = f"{a}, {b}"
3      # return a
4      return a * b
5
6  example = func(5, 4.5)
7  print(example)
```

# Типизация

- Помимо этого, можно параметризовать более сложные типы, например, `List`. Такие типы могут принимать значения параметров, которые помогают более точно описать тип функции. Так, например, `List[int]` указывает на то, что список состоит только из целочисленных значений.

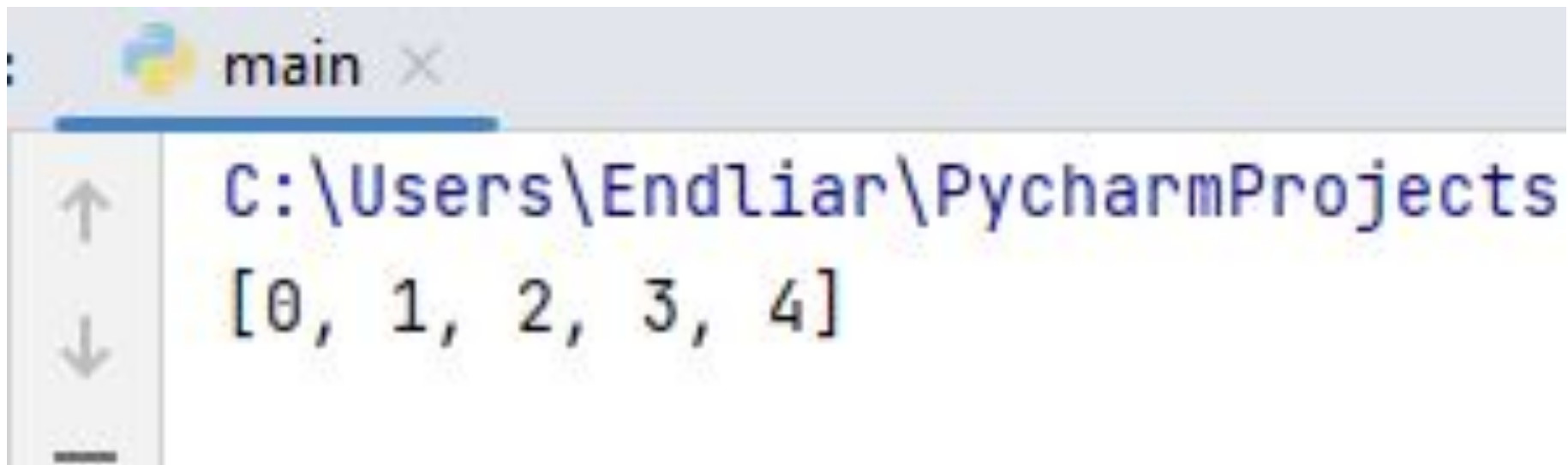
# Типизация

- Пример кода:

```
main.py x
1  from typing import List
2
3  def func(n: int) -> List[int]:
4      return list(range(n))
5
6  example = func(5)
7  print(example)
```

# Типизация

- Вывод:



```
main x
C:\Users\Endliar\PycharmProjects
[0, 1, 2, 3, 4]
```

The image shows a screenshot of a Python terminal window. The window title is "main x". The terminal output consists of two lines: the first line is the file path "C:\Users\Endliar\PycharmProjects" and the second line is the list "[0, 1, 2, 3, 4]". The terminal has a vertical scrollbar on the left side with up, down, and home icons.



# Типизация

- Кроме List, существуют и другие типы из модуля typing, которые можно параметризовать. Такие типы называются Generic-типами. Подробнее о них можете погуглить, и посмотреть по официальной (или не очень) документации. Их вполне приемлемое количество на разный цвет и вкус.

# Типизация

- При этом функции тоже имеют свои типы. Например, для описания функции можно использовать тип `Callable`, где указываются типы входных параметров и возвращаемых значений. Пример использования представлен ниже:

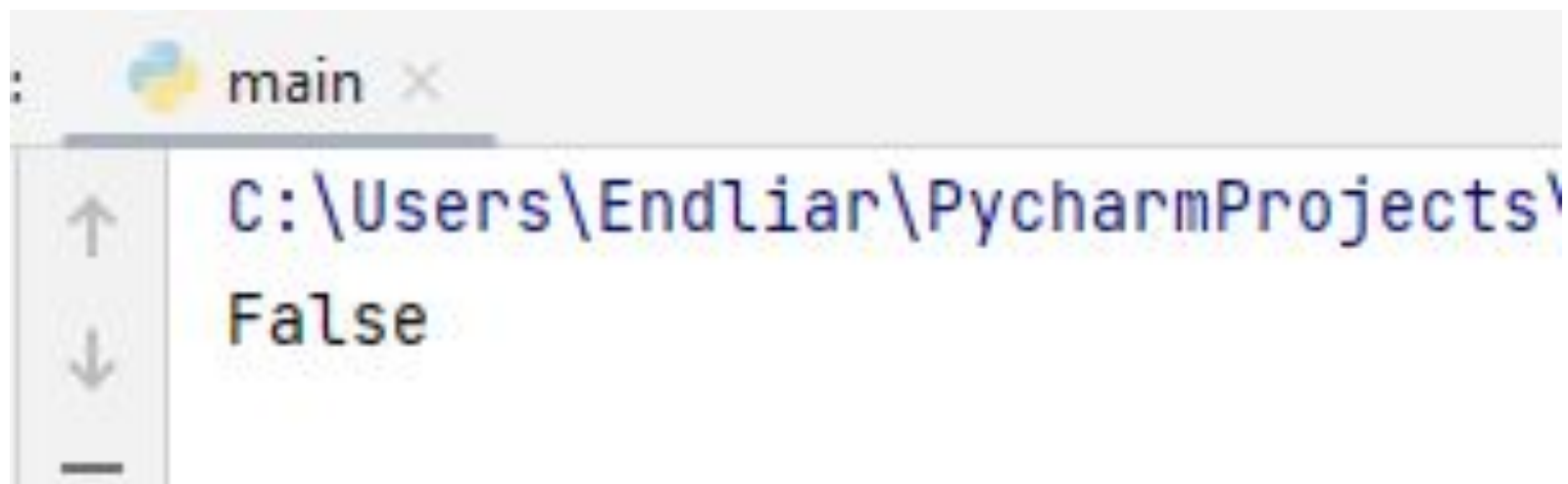
# Типизация

- Пример:

```
3
4  from typing import Callable
5
6  def func(f: Callable[[int, int], bool]) -> bool:
7      return f(5, 6)
8
9  example = func(lambda x, y: x == y)
10 print(example)
```

# Типизация

- Вывод:



```
main x
C:\Users\Endliar\PycharmProjects\
False
```

# Lambda выражение в Python

- lambda оператор или lambda функция в Python это способ создать анонимную функцию, то есть функцию без имени. Такие функции можно назвать одноразовыми, они используются только при создании. Как правило, lambda функции используются в комбинации с функциями filter, map, reduce.

# Типизация

- Тип Callable говорит о том, что:
  1. У объекта реализован метод `__call__`.
  2. Описывает типы параметров этому методу.
- На первом месте стоит массив типов входных параметров, на втором — тип возвращаемого значения.
- Про остальные абстрактные типы контейнеров можно прочитать в документации Python.

# Основной смысл

- Цель — указать разработчику на ожидаемый тип данных при получении или возврате данных из функции или метода. В свою очередь, это позволяет сократить количество багов, ускорить написание кода и улучшить его качество.

# ОСНОВНОЙ СМЫСЛ

```
21
22 a: List[Dict[str, List[ReturnTemplate]]] = []
23
24 a[0].
25 items
26 clear
27 copy
28 fromkeys
29 get
30 keys
31 pop
32 popitem
33 setdefault
   update
   values
   __annotations__
```

items: () -> dict\_items[str, List[ReturnTemplate]]

D.items() -> a set-like object providing a view on D's items

```
22 a = []
23
24 a[0].
25 Предложения отсутствуют.
```