Python

09

Документирование кода

https://numpydoc.readthedocs.io/en/latest/format.html

https://habr.com/ru/company/lamoda/blog/432656/

Парадигма — это стиль написания исходного кода компьютерной программы.

Самые популярные парадигмы

- Императивное программирование
- Структурное программирование
- Декларативное программирование
- Объектно-ориентированное программирование

Императивное программирование

Императивное программирование (от англ. imperative — приказ) — это парадигма программирования, которая описывает процесс вычисления в виде инструкций, **изменяющих состояние данных**. Мы описываем **КАК** выполнить задачу.

Для этой парадигмы характерно использование:

- именованных переменных;
- оператора присваивания;
- составных выражений;
- подпрограмм;
- циклов.

К подвидам императивного программирования относят **Процедурное** и **Объектно-ориентированное программирование (ООП)**.

Декларативное программирование

Декларативное программирование — это парадигма программирования, в которой задается спецификация решения задачи, то есть описывается, **ЧТО** представляет собой проблема и ожидаемый результат. Декларативные программы **не используют состояния**, то есть **не содержат переменных и операторов** присваивания. Программа — спецификация описывающая решение задачи.

К подвидам декларативного программирования относят **Функциональное** и **Логическое программирование**.

Функциональное программирование

Функциональное программирование — программирование значениями (не используются присваивания). Предполагает обходиться вычислением **результатов функций** от исходных данных и результатов других функций, и **не предполагает явного хранения состояния**.

Для этой парадигмы характерно:

- функции первого класса (можно передавать как аргументы и возвращать из других функций);
- функции высшего порядка (принимают на вход другие функции);
- рекурсии;
- состояние никогда не меняется;
- не используется присваивание.

Основой для функционального программирования являются **Лямбда-исчисления**, многие функциональные языки можно рассматривать как «надстройку» над ними.

Лямбда-функции — это функции, у которой фактически нет имени. Лямда-выражения — анонимные функции.

lamda выражения

```
func = lambda x, y, z: x + y + z
```

Задание 9.01

Создать lambda функцию, которая принимает на вход имя и выводит его в формате “Hello, {name}”

Задание 9.02

Создать lambda функцию, которая принимает на вход список имен и выводит их в формате “Hello, {name}” в другой список

map()

```
result = map(lambda x: x ** 2,
[1,2,3,4,5,6])


print(list(result))
```

Задание 9.03

Дан список чисел. Вернуть список, где каждый число переведено в строку

[5, 3] -> [‘5’, ‘3’]

filter()

```
result = filter(lambda x: x % 2 == 0,
[1,2,3,4,5,6])


print(list(result))
```

Задание 9.04

Дан список имен, отфильтровать все имена, где есть буква k

reduce()

```
from functools import reduce
result = reduce(lambda a, x: a + x ** 2, [1,2,3], 0)
```

Задание 9.05

Дан список чисел. Найти произведение всех чисел, которые кратны 3.

decorators

```
def my_decorator(func):
  def do_some_staff():
      # some action
      result = func()
      # some action
      return result
  return do_some_staff
def my_func():
  pass
my_new_func = my_decorator(my_func)
my_new_func()
```

decorators

```
def my_decorator(func):
  def do_some_staff():
      # some action
      result = func()
      # some action
      return result
  return do_some_staff
@my_decorator
def my_func():
  pass
my_func()
```

Задание 9.06

Написать декоратор, который будет выводить время выполнения функции

```
from datetime import datetime

time = datetime.now()
```