

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития Кафедра
инфокоммуникаций

Дисциплина «Языки программирования»

Отчет по практической работе № 2.12

Декораторы функций в языке Python

Выполнил: студент группы ИТС-б-о-21-1
Крамаренко Илья Витальевич

(подпись)

Проверил: к.т.н., доцент Кафедры
инфокоммуникаций Воронкин
Р.А.

(подпись)

Ставрополь, 2022

Декораторы функций в языке Python

Цель работы: приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.

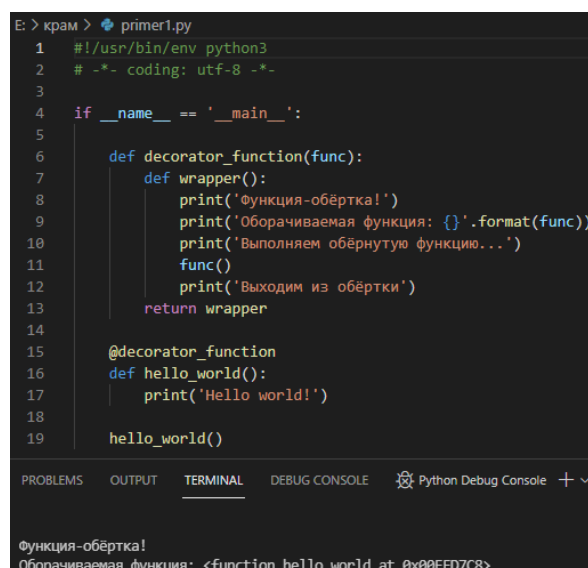
Теоретический материал:

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. Вот почему декораторы можно рассматривать как практику метапрограммирования, когда программы могут работать с другими программами как со своими данными. Чтобы понять, как это работает, сначала разберёмся в работе функций в Python.

Процедура — это именованная последовательность вычислительных шагов. Любую процедуру можно вызвать в любом месте программы, в том числе внутри другой процедуры или даже самой себя.

Ход работы:

Выполненные примеры:



```
E: > крэм > primer1.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5
6      def decorator_function(func):
7          def wrapper():
8              print('Функция-обёртка!')
9              print('Оборачиваемая функция: {}'.format(func))
10             print('Выполняем обернутую функцию...')
11             func()
12             print('Выходим из обёртки')
13             return wrapper
14
15         @decorator_function
16         def hello_world():
17             print('Hello world!')
18
19         hello_world()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python Debug Console + v

Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x00EFD7C8>

Рисунок 1. Пример №1

```
E: > крэм > primer2.py > ...
1  env python3
2  g: utf-8 -*-
3
4  == '__main__':
5
6
7  chmark(func):
8  ort time
9
10  wrapper():
11  start = time.time()
12  func()
13  end = time.time()
14  print('[*] Время выполнения: {} секунд.'.format(end-start))
15  urn wrapper
16
17  ark
18  ch_webpage():
19  ort requests
20  page = requests.get('https://google.com')
21
22  ebpape()
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE Python Debug Console + v

scode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\de
auncher' '63452' '--' 'e:\крэм\primer2.py'

Traceback (most recent call last):
File "e:\крэм\primer2.py", line 22, in <module>
fetch_webpage()
File "e:\крэм\primer2.py", line 12, in wrapper
func()
File "e:\крэм\primer2.py", line 19, in fetch_webpage
import requests

Рисунок 2. Пример №2

```
E: > крэм > primer3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5
6
7      def benchmark(func):
8          import time
9
10         def wrapper(*args, **kwargs):
11             start = time.time()
12             return_value = func(*args, **kwargs)
13             end = time.time()
14             print('[*] Время выполнения: {} секунд.'.format(end-start))
15             return return_value
16         return wrapper
17
18     @benchmark
19     def fetch_webpage(url):
20         import requests
21         webpage = requests.get(url)
22         return webpage.text
23
24     webpage = fetch_webpage('https://google.com')
25     print(webpage)
```

Рисунок 3. Пример №3

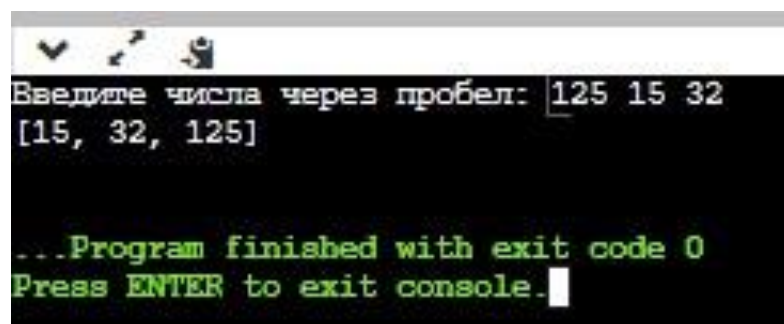
Вариант №2. Индивидуальное задание:

2. На вход программы поступает строка из целых чисел, записанных через пробел. Напишите функцию `get_list`, которая преобразовывает эту строку в список из целых чисел и возвращает его. Определите декоратор для этой функции, который сортирует список чисел, полученный из вызываемой в нем функции. Результат сортировки должен возвращаться при вызове декоратора. Вызовите декорированную функцию `get_list` и отобразите полученный отсортированный список на экране.

Рисунок 4. Вариант для выполнения индивидуального задания

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # На вход программы поступает строка из целых чисел, записанных через пробел. Напишите
5  # функцию get_list, которая преобразовывает эту строку в список из целых чисел и
6  # возвращает его. Определите декоратор для этой функции, который сортирует список чисел,
7  # полученный из вызываемой в нем функции. Результат сортировки должен возвращаться при
8  # вызове декоратора. Вызовите декорированную функцию get_list и отобразите полученный
9  # отсортированный список на экране.
10
11 def list_sort(func):
12     def inner(s):
13         return sorted(func(s))
14     return inner
15
16 @list_sort
17 def get_list(s):
18     return [int(i) for i in s.split()]
19
20 if __name__ == '__main__':
21     print(get_list(input('Введите числа через пробел: ')))
```

Рисунок 5. Написанный код для выполнения задания



```
Введите числа через пробел: 125 15 32
[15, 32, 125]

...Program finished with exit code 0
Press ENTER to exit console.
```

Рисунок 6. Результат написанного кода

Контрольные вопросы:

1. Что такое декоратор?

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса?

Объектами первого класса в контексте конкретного языка программирования называются элементы, с которыми можно делать всё то же, что и с любым другим объектом: передавать как параметр, возвращать из функции и присваивать переменной. В Python всё является объектом, а не только объекты, которые вы создаёте из классов. Это значит, что в Python всё это — объекты:

- числа;
- строки;
- классы;
- функции.

3. Каково назначение функций высших порядков?

Функции высших порядков — это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Декоратор — это функция, которая позволяет обернуть другую функцию с помощью символа «@»

5. Какова структура декоратора функций?

Сначала записывается функция — декоратор. Потом идет его вызов с помощью @, а затем основная функция, которую оборачивает декоратор.

6. Самостоятельно изучить, как можно передать параметры декоратору, а не декорируемой функции?

Используя замыкание функций.

Вывод: в ходе лабораторной работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.