Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.12**

**дисциплины «Программирование на Python»**

**Вариант\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Иващенко Олег Андреевич  2 курс, группа ИВТ-б-о-22-1,  09.03.02 «Информационные и вычислительные машины», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Роман Александрович, доцент кафедры инфокоммуникаций  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Тема**: «Декораторы функций в языке Python»

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

Порядок выполнения работы

Таблица 1 – Код программы example.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  def benchmark(func):  import time    def wrapper():  start = time.time()  func()  end = time.time()  print(f"Время выполнения: {end - start} сек.")  return wrapper  @benchmark  def fetch\_webpage():  import requests  webpage = requests.get('https://www.google.com')  if \_\_name\_\_ == "\_\_main\_\_":  fetch\_webpage() |



Рисунок 1 – Результат выполнения программы

Индивидуальное задание. Вводятся два списка (каждый с новой строки) из слов, записанных через пробел. Имеется функция, которая преобразовывает эти две строки в два списка слов и возвращает эти списки. Определите декоратор для этой функции, которой из этих двух списков формирует словарь, в котором ключами являются слова из первого списка, а значениями – соответствующие элементы из второго списка. Полученный словарь должен возвращаться при вызове декоратора. Примените декоратор к первой функции и вызовите её. Результат (словарь) отобразите на экране.

Таблица 2 – Код программы individual.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  def dict\_transform(func):  def wrapper(list1, list2):  dictionary = func(list1, list2)  return dict(zip(dictionary[0], dictionary[1]))  return wrapper  @dict\_transform  def print\_dictionary(list1, list2):  return list1, list2  if \_\_name\_\_ == "\_\_main\_\_":  list1 = input("Введите первую строку: ").split()  list2 = input("Введите вторую строку: ").split()  result = print\_dictionary(list1, list2)  print(f"Результат: {result}") |



Рисунок 2 – Результат выполнения программы

Контрольные вопросы

1. Что такое декоратор?

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

1. Почему функции являются объектами первого класса?

В Python функции считаются объектами первого класса, потому что они могут быть переданы как аргументы в другие функции, возвращены из функций, присвоены переменным и сохранены в структурах данных.

1. Каково назначение функций высших порядков?

Функции высших порядков в Python – это функции, которые принимают одну или несколько функций в качестве аргументов, либо возвращают другую функцию. Они позволяют абстрагироваться от конкретных действий, делегируя функциональность более обобщёнными функциями.

1. Как работают декораторы?

Декораторы работают, оборачивая функцию другой функцией. Когда применяется декоратор к функции, он заменяет или модифицирует поведение этой функции. Это обеспечивает прозрачное расширение или изменение функциональности без изменения самой функции.

1. Какова структура декоратора функций?

Декоратор функций в Python обычно представляет собой функцию, которая принимает одну функцию в качестве аргумента и возвращает другую функцию.

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

Декоратор может принимать параметры, используя дополнительный уровень вложенности. Например:

def my\_decorator(arg1, arg2):

def decorator(func):

def wrapper(\*args, \*\*kwargs):

print(f”Декоратор получил аргументы: {arg1}, {arg2}”)

func(\*args, \*\*kwargs)

return wrapper

return decorator

@my\_decorator(“Аргумент 1”, “Аргумент 2”)

def say\_hello(name):

print(f”Привет, {name}”)

say\_hello(“Иван”)

В этом примере my\_decorator принимает два аргумента и возвращает сам декоратор, который в свою очередь оборачивает функцию. При использовании декоратора с аргументами, их можно передавать как обычные аргументы при применении декоратора к функции.

**Выводы**: В процессе выполнения лабораторной работы были приобретены навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x, был разобран пример, код которого представлен в таблице 1 и файле example.py данного репозитория, а так же было выполнено индивидуальное задание, код которого представлен в таблице 2 и файле individual.py данного репозитория.