

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №2.8

**Дисциплина: «Программирование на Python»**  
**Тема: «Работа с функциями в языке Python»**

Выполнил:  
Епифанов Алексей Александрович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных систем  
», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович

---

(подпись)

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

Ставрополь, 2023 г.

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Проработал пример лабораторной работы:

```
aleksejerifanov@MacBook-Pro-Aleksej program % ./prim.py
>>> add
Фамилия и инициалы? Епифанов А. А.
Должность? Студент
Год поступления? 2022
>>> list
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Епифанов А. А. | Студент | 2022 |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Absd A. A.
Должность? Prepodavatel
Год поступления? 2003
>>> list
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Absd A. A. | Prepodavatel | 2003 |
| 2 | Епифанов А. А. | Студент | 2022 |
+-----+-----+-----+-----+
>>> help
Список команд:

add – добавить работника;
list – вывести список работников;
select <стаж> – запросить работников со стажем;
help – отобразить справку;
exit – завершить работу с программой.
>>> select 1
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Absd A. A. | Prepodavatel | 2003 |
| 2 | Епифанов А. А. | Студент | 2022 |
+-----+-----+-----+-----+
>>> select 13
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Absd A. A. | Prepodavatel | 2003 |
+-----+-----+-----+-----+
>>> exit
aleksejerifanov@MacBook-Pro-Aleksej program %
```

Рисунок 1. Запуск программы примера

3. Решил задачу 1: Решить следующую задачу: основная ветка программы, не считая заголовков функций, состоит из двух строк кода. Это вызов функции `test()` и инструкции. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное".

Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def test():
    """
    Запросить число и вызвать соответствующую функцию.
    Вызывается positive() если число положительное.
    Вызывается negative() если отрицательное.
    Возвращает None если == 0.
    """
    number = int(input("Введите целое число: "))
    if number > 0:
        positive()
    elif number < 0:
        negative()
    else:
        return
def positive():
    """
    Вывести на экран 'Положительное'.
    """
    print("Положительное")
def negative():
    """
    Вывести на экран 'Отрицательное'.
    """
    print("Отрицательное")
if __name__ == '__main__':
```

```
test()
aleksej@epifanov MacBook-Pro-Aleksej program % ./prim.py
>>> add
Фамилия и инициалы? Епифанов А. А.
Должность? Студент
Год поступления? 2022
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Епифанов А. А.          |      Студент        |      2022     |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Absd А. А.
Должность? Prepodavatel
Год поступления? 2003
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Absd А. А.              | Prepodavatel        |      2003     |
|  2 | Епифанов А. А.          |      Студент        |      2022     |
+-----+-----+-----+-----+
>>> help
Список команд:

add – добавить работника;
list – вывести список работников;
select <стаж> – запросить работников со стажем;
help – отобразить справку;
exit – завершить работу с программой.
>>> select 1
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Absd А. А.              | Prepodavatel        |      2003     |
|  2 | Епифанов А. А.          |      Студент        |      2022     |
+-----+-----+-----+-----+
>>> select 13
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Absd А. А.              | Prepodavatel        |      2003     |
+-----+-----+-----+-----+
>>> exit
aleksej@epifanov MacBook-Pro-Aleksej program %
```

Рисунок 2. Вывод программы task1

4. Решил задачу 2: Решите следующую задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле  $S = \pi R^2$ . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле  $S_{\text{бок}} = 2\pi R H$ , или полную площадь цилиндра. В последнем случае

к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции circle().

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import math
def cylinder():
    """
    Вычисляет полную площадь цилиндра или только боковой поверхности.
    При выборе полной площади вызывается дополнительно функция circle().
    """
    def circle(radius):
        """
        Вычисляет площадь круга по радиусу.
        Принимает радиус круга и возвращает площадь, используя формулу
        площади круга:  $\pi * r^2$ .
        """
        return math.pi * radius ** 2
    radius = float(input("Введите радиус цилиндра: "))
    height = float(input("Введите высоту цилиндра: "))
    side_area = 2 * math.pi * radius * height
    sw = True
    while sw:
        sw = False
        user_choice = input(
            "Введите 'боковая' для площади боковой поверхности" +
            " или 'полная' для полной площади цилиндра: "
        )
        match user_choice.lower():
            case 'боковая':
                print(f"Площадь боковой поверхности цилиндра: {side_area:.2f}")
            case 'полная':
                base_area = circle(radius)
                full_area = side_area + 2 * base_area
                print(f"Полная площадь цилиндра: {full_area:.2f}")
            case _:
                print(
                    "\n\nНеправильный ввод. Пожалуйста, введите 'боковая' или
'полная'.\n\n")
                sw = True
if __name__ == '__main__':
    cylinder()
```

```

● aleksejefifanov@MacBook-Pro-Aleksej program % ./task2.py
Введите радиус цилиндра: 23
Введите высоту цилиндра: 4
Введите 'боковая' для площади боковой поверхности или 'полная' для полной площади цилиндра: боковая
Площадь боковой поверхности цилиндра: 578.05
● aleksejefifanov@MacBook-Pro-Aleksej program % ./task2.py
Введите радиус цилиндра: 23
Введите высоту цилиндра: 4
Введите 'боковая' для площади боковой поверхности или 'полная' для полной площади цилиндра: полная
Полная площадь цилиндра: 3901.86
● aleksejefifanov@MacBook-Pro-Aleksej program % ./task2.py
Введите радиус цилиндра: 2
Введите высоту цилиндра: 4
Введите 'боковая' для площади боковой поверхности или 'полная' для полной площади цилиндра: боковая
Площадь боковой поверхности цилиндра: 50.27
● aleksejefifanov@MacBook-Pro-Aleksej program % ./task2.py
Введите радиус цилиндра: 2
Введите высоту цилиндра: 4
Введите 'боковая' для площади боковой поверхности или 'полная' для полной площади цилиндра: полная
Полная площадь цилиндра: 75.40
○ aleksejefifanov@MacBook-Pro-Aleksej program % |

```

Рисунок 3. Вывод программы task2

5. Решил задачу 3: Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Решите следующую задачу: напишите функцию, которая считывает
# с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0.
# Функция должна возвращать полученное произведение. Вызовите функцию
# и выведите на экран результат ее работы.
def multiply_numbers():
    """
    Функция считывает числа с клавиатуры и перемножает их.
    Функция запрашивает числа до тех пор, пока не будет введен 0.
    Возвращает полученное произведение.
    """
    result = 1
    while True:
        number = int(input("Введите число:"))
        if number == 0:
            break
        result *= number
    return result
if __name__ == "__main__":
    result = multiply_numbers()
    print("Результат:", result)

```

```

● aleksejerifanov@MacBook-Pro-Aleksej program % ./task3.py
Введите число:5
Введите число:4
Введите число:3
Введите число:2
Введите число:1
Введите число:0
Результат: 120
● aleksejerifanov@MacBook-Pro-Aleksej program % ./task3.py
Введите число:1
Введите число:1
Введите число:1
Введите число:0
Результат: 1
● aleksejerifanov@MacBook-Pro-Aleksej program % ./task3.py
Введите число:3
Введите число:3
Введите число:3
Введите число:3
Введите число:3
Введите число:3
Введите число:0
Результат: 243
○ aleksejerifanov@MacBook-Pro-Aleksej program % |

```

Рисунок 4. Вывод программы task3

6. Решил задачу 4: Решите следующую задачу: напишите программу, в которой определены следующие четыре функции: Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

```
#!/usr/bin/env python3
```

```

# -*- coding: utf-8 -*-
def get_input():
    """
    Функция запрашивает ввод с клавиатуры и возвращает введенную строку.
    """
    return input("Введите значение: ")
def test_input(value):
    """
    Функция проверяет, можно ли преобразовать переданное значение в целое
    число.
    Возвращает True, если преобразование возможно, иначе False.
    """
    return (value.isdigit() or value[0] == '-')
def str_to_int(value):
    """
    Функция преобразовывает переданное значение в целочисленный тип.
    Возвращает полученное число.
    """
    return int(value)
def print_int(value):
    """
    Функция выводит переданное значение на экран.
    """
    print(value)
if __name__ == "__main__":
    input_value = get_input()
    if test_input(input_value):
        int_value = str_to_int(input_value)
        print_int(int_value)
    else:
        print("Значение не может быть преобразовано в целое число")

```



```

● aleksejeripfanov@MacBook-Pro-Aleksej program % ./task4.py
Введите значение: 45
45
● aleksejeripfanov@MacBook-Pro-Aleksej program % ./task4.py
Введите значение: -32
-32
● aleksejeripfanov@MacBook-Pro-Aleksej program % ./task4.py
Введите значение: 2.4
Значение не может быть преобразовано в целое число
● aleksejeripfanov@MacBook-Pro-Aleksej program % ./task4.py
Введите значение: 3,4
Значение не может быть преобразовано в целое число
● aleksejeripfanov@MacBook-Pro-Aleksej program % ./task4.py
Введите значение: tr3
Значение не может быть преобразовано в целое число
○ aleksejeripfanov@MacBook-Pro-Aleksej program % |

```

Рисунок 5. Вывод программы task5

7. Выполнил индивидуальное задание вариант 10: Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import bisect
import re
import sys
def get_route():
    """
    Запросить данные о маршруте.
    """
    start = input("Введите начальный пункт: ")
    end = input("Введите конечный пункт: ")
    count = int(input("Введите номер маршрута: "))
    return {
        'начальный пункт': start,
        'конечный пункт': end,
        'номер маршрута': count
    }
def display_routes(routes):
    """
    Отобразить список маршрутов.

```

```

"""
if routes:
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^30} | {:^20} | {:^8} |'.format(
            "Начало",
            "Конец",
            "Номер"
        )
    )
    print(line)
    for route in routes:
        print(
            '| {:<30} | {:<20} | {:>8} |'.format(
                route.get('начальный пункт', ''),
                route.get('конечный пункт', ''),
                route.get('номер маршрута', '')
            )
        )
    print(line)
else:
    print("Список маршрутов пуст.")
def select_routes(routes, name_punct):
    """
    Выбрать маршруты с заданным пунктом отправления или прибытия.
    """
    selected = []
    for route in routes:
        if route['начальный пункт'].lower() == name_punct \
            or route['конечный пункт'].lower() == name_punct:
            selected.append(route)
    return selected
def main():
    """
    Главная функция программы.
    """
    routes = []
    while True:
        command = input(">>> ").lower()
        match command:

```

```

case 'exit':
    break
case 'add':
    route = get_route()
    if route not in routes:
        bisect.insort(
            routes, route, key=lambda item: item.get('номер маршрута'))
    else:
        print("Данный маршрут уже добавлен.")
case 'list':
    display_routes(routes)
case _ if (m := re.match(r'select (.+)', command)):
    name_punct = m.group(1)
    selected = select_routes(routes, name_punct)
    display_routes(selected)
case 'help':
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <название пункта> - запросить маршруты, которые
начинаются\n"
        "или заканчиваются в данном пункте;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
case _:
    print(f"Неизвестная команда {command}", file=sys.stderr)
if __name__ == '__main__':
    main()

```

```

● aleksejefifanov@MacBook-Pro-Aleksej program % ./ind.py
>>> add
Введите начальный пункт: Россия
Введите конечный пункт: Китай
Введите номер маршрута: 23
>>> add
Введите начальный пункт: Россия
Введите конечный пункт: Португалия
Введите номер маршрута: 41
>>> add
Введите начальный пункт: Португалия
Введите конечный пункт: Китай
Введите номер маршрута: 12
>>> list
+-----+-----+-----+
|          Начало          |          Конец          |          Номер          |
+-----+-----+-----+
| Португалия                | Китай                   |          12             |
| Россия                    | Китай                   |          23             |
| Россия                    | Португалия              |          41             |
+-----+-----+-----+
>>> select Россия
+-----+-----+-----+
|          Начало          |          Конец          |          Номер          |
+-----+-----+-----+
| Россия                    | Китай                   |          23             |
| Россия                    | Португалия              |          41             |
+-----+-----+-----+
>>> select Португалия
+-----+-----+-----+
|          Начало          |          Конец          |          Номер          |
+-----+-----+-----+
| Португалия                | Китай                   |          12             |
| Россия                    | Португалия              |          41             |
+-----+-----+-----+
>>> select Китай
+-----+-----+-----+
|          Начало          |          Конец          |          Номер          |
+-----+-----+-----+
| Португалия                | Китай                   |          12             |
| Россия                    | Китай                   |          23             |
+-----+-----+-----+
>>> help
Список команд:

add – добавить маршрут;
list – вывести список маршрутов;
select <название пункта> – запросить маршруты, которые начинаются
или заканчиваются в данном пункте;
help – отобразить справку;
exit – завершить работу с программой.
>>> exit
○ aleksejefifanov@MacBook-Pro-Aleksej program % |

```

Рисунок 6. Запуск программы индивидуального задания

Ответы на контрольные вопросы:

### 1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

### 2. Каково назначение операторов `def` и `return` ?

Ключевое слово `def` сообщает интерпретатору, что перед ним определение функции. За `def` следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена.

Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`.

Если интерпретатор Питона, выполняя тело функции, встречает `return`, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

### 3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей

программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение. К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

#### 4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.

Фокус здесь в том, что перечисление значений через запятую (например, 10, 15, 19 ) создает объект типа `tuple`.

#### 5. Какие существуют способы передачи значений в функцию?

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым.

Параметры представляют собой локальные переменные, которым присваиваются значения в момент вызова функции. Конкретные значения, которые передаются в функцию при ее вызове, будем называть аргументами. Следует иметь в виду, что встречается иная терминология. Например, формальные параметры и фактические параметры. В Python же обычно все называют аргументами.

Когда интерпретатор переходит к функции, чтобы начать ее исполнение, он присваивает переменным-параметрам переданные в функцию значения-аргументы.

Изменение значений переменных в теле функции никак не скажется на значениях, переданных в нее. Они останутся прежними. В Python такое поведение характерно для неизменяемых типов данных, к которым относятся,

например, числа и строки. Говорят, что в функцию данные передаются по значению.

Существуют изменяемые типы данных. Для Питона, это, например, списки и словари. В этом случае данные передаются по ссылке. В функцию передается ссылка на них, а не сами данные. И эта ссылка связывается с локальной переменной. Изменения таких данных через локальную переменную обнаруживаются при обращении к ним через глобальную. Это есть следствие того, что несколько переменных ссылаются на одни и те же данные, на одну и ту же область памяти.

#### 6. Как задать значение аргументов функции по умолчанию?

В Python у функций бывают параметры, которым уже присвоено значение по умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать. Тогда значение по умолчанию заменится на переданное.

Например, в следующем определении функции параметр со значением по умолчанию будет `r`: `def cylinder(h, r=1)`.

#### 7. Каково назначение `lambda`-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые `lambda`-функции могут быть использованы везде, где требуется функция.

`lambda` – это выражение, а не инструкция. По этой причине ключевое слово `lambda` может появляться там, где синтаксис языка Python не позволяет использовать инструкцию `def`, –внутри литералов или в вызовах функций, например.

#### 8. Как осуществляется документирование кода согласно PEP257?

PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в

себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода. Такая строка документации становится специальным атрибутом `__doc__` этого объекта.

Все модули должны, как правило, иметь строки документации, и все функции и классы, экспортируемые модулем, также должны иметь строки документации. Публичные методы (в том числе `__init__`) также должны иметь строки документации. Пакет модулей может быть документирован в `__init__.py`.

Для согласованности, всегда нужно использовать `"""triple double quotes"""` для строк документации. Можно использовать `r"""raw triple double quotes"""`, если удет присутствовать обратная косая черта в строке документации. Существует две формы строк документации: однострочная и многострочная.

9. В чем особенность однострочных и многострочных форм строк документации?

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке.

Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции).

Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем интроспекции.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими средствами индексации, поэтому важно, чтобы она находилась на одной



строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке

Вывод: в результате выполнения работы были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.