

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 РАБОТЕ №2.8**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:  
Мурашко Анастасия Юрьевна  
1 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. тех. наук, доцент,  
доцент кафедры инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

**Тема:** работа с функциями в языке Python.

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

Порядок выполнения работы:

### Пример 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import ...

def get_worker()::
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))

    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }

def display_workers(staff)::
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
```

### Рисунок 1 – Пример

```
        "Должность",
        "Год"
    )
    )
    print(line)

    # Вывести данные о всех сотрудниках.
    for idx, worker in enumerate(staff, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                idx,
                worker.get('name', ''),
                worker.get('post', ''),
                worker.get('year', 0)
            )
        )
        print(line)

    else:
        print("Список работников пуст.")

def select_workers(staff, period)::
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()

    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)

    # Возвратить список выбранных работников.
    return result
```

### Рисунок 2 – Пример

```

def main():
    """
    Главная функция программы.
    """

    # Список работников.
    workers = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.
            worker = get_worker()

            # Добавить словарь в список.
            workers.append(worker)

            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            # Отобразить всех работников.
            display_workers(workers)

        elif command.startswith('select '):
            # Разбить команду на части для выделения стажа.
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])

            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(selected)

```

Рисунок 3 – Пример

```

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("\nСписок команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Рисунок 4 – Пример

## Общее задание 1.

Добавила новый файл *Obsheel.py*

*Условие задания:* основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. 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 -*-

number = int(input("Введите отрицательное или положительное число: "))

def positive():
    print("Положительное")

def negative():
    print("Отрицательное")

def test():
    if number >= 0:
        positive()
    elif number < 0:
        negative()

if __name__ == '__main__':
    test()
```



Рисунок 5 – Результат программы общего задания №1

Порядок определения функций не имеет значения, поэтому функции `positive()` и `negative()` могут следовать после вызова функции `test()`. Это связано с тем, что в Python функции создаются целиком при первом выполнении программы, а не в момент определения. При вызове функции `test()` все функции уже будут определены и могут быть использованы.

## Общее задание 2.

Добавила новый файл *Obshee2.py*

*Условие задания:* 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()`.

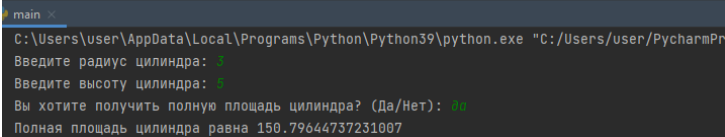
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math

def circle(radius):
    return math.pi * radius**2

def cylinder():
    radius = float(input("Введите радиус цилиндра: "))
    height = float(input("Введите высоту цилиндра: "))
    side = 2 * math.pi * radius * height
    command = input("Вы хотите получить полную площадь цилиндра? (Да/Нет): ")
    if command.lower() == 'да':
        circle_area = 2 * circle(radius)
        full_area = side + circle_area
        print("Полная площадь цилиндра равна", full_area)
    else:
        print("Площадь боковой поверхности цилиндра равна", side)

if __name__ == '__main__':
    cylinder()
```



```
main
C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/user/PycharmPro
Введите радиус цилиндра: 3
Введите высоту цилиндра: 5
Вы хотите получить полную площадь цилиндра? (Да/Нет): да
Полная площадь цилиндра равна 150.79644737231007
```

Рисунок 6 – Результат программы общего задания №2

## Общее задание 2.

Добавила новый файл *Obshee3.py*

*Условие задания:* напишите программу, в которой определены следующиечетыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры ивозвращает 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`.

2. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

3. Функция `print_int()` имеет один параметр. Она выводит переданное значениена 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():
    return input("Введите значение: ")

def test_input(value):
    try:
        int(value)
        return True
    except ValueError:
        return False

def str_to_int(value):
    return int(value)

def print_int(value):
    print(f"Введенное значение: {value}")

# Основная ветка программы
if __name__ == '__main__':
    value = get_input()
    if test_input(value):
        value_int = str_to_int(value)
        print_int(value_int)
    else:
        print("Ошибка: введенное значение не является целым числом")
```

main x

C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:/Use

Введите значение: 32

Введенное значение: 32

Рисунок 7 – Результат программы общего задания

### Индивидуальное задание.

Создала файл под названием *idz.py*

Оформить в виде функций индивидуальное задание из лабораторной работы №2.6.

*Условие задание (2.6):* Использовать словарь, содержащий следующие ключи: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршрутах, которые начинаются

или оканчиваются в пункте, название которого введено с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Та же программа только в виде функций.

```
import sys

list_route = []
spisok_new = []

def add_route():
    name_start = input('Начало маршрута: ')
    name_finish = input('Конец маршрута: ')
    num_route_get = input('Номер маршрута: ')

    num_route = int(num_route_get)

    if type(num_route) != int:
        print("Введенные данные не верны!")

    list_route_new = {
        'name_start': name_start,
        'name_finish': name_finish,
        'num_route': num_route
    }

    list_route.append(list_route_new)

    if len(list_route) > 1:
        list_route.sort(key=lambda item: item.get('num_route', ''))

def print_routes():
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 15,
        '-' * 30,
```



```
for idx, spisok_new in enumerate(list_route, 1):
    print(
        '| {:>4} | {:<15} | {:<30} | {:<20} | '.format(
            idx,
            spisok_new.get('name_start', ''),
            spisok_new.get('name_finish', ''),
            spisok_new.get('num_route', 0)
        )
    )
```

main x

```
>>> list
```

№	Начало маршрута	Конец маршрута	№ Маршрута
1	rjirf	asd	2

```
>>>
```

Рисунок 7 – Результат.

### Контрольные вопросы.

1) Каково назначение функций в языке программирования Python?

Функции в языке программирования Python используются для группировки набора инструкций в одну логическую единицу, которая может быть вызвана из любой части программы. Использование функций позволяет избежать повторного кода, улучшить читаемость программы, сделать код более организованным и легким для тестирования и сопровождения. Также функции могут быть использованы для создания модулей, которые могут быть использованы в различных проектах.

2) Каково назначение операторов def и return?

Оператор `def` в языке программирования Python используется для определения функции. Функция представляет собой блок кода с именем, который может вызываться из других частей программы. Оператор `return` внутри функции позволяет вернуть значение из функции.

Проще говоря, оператор `def` используется для создания функций, а оператор `return` - для возвращения результатов работы функции, которые могут быть использованы в другой части программы. Функции позволяют разбить код на более мелкие логические блоки, повторно использовать код, упрощать чтение и понимание программы в целом.

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

Локальные переменные - это переменные, которые определены внутри функции и могут быть использованы только внутри этой функции. Они не видны за ее пределами и существуют только во время выполнения функции.

Глобальные переменные - это переменные, которые определены вне функции и могут быть использованы в любой части программы, включая функции. Они существуют в течение всего времени работы программы.

Назначение локальных переменных - это изолировать код функции от других частей программы, чтобы избежать изменений переменных из других частей программы, которые могут негативно повлиять на работу функции.

Назначение глобальных переменных - это обеспечить доступ к данному объекту из любой части программы. Однако, существует опасность перезаписи глобальных переменных, и использование глобальных переменных следует использовать с осторожностью.

4) Как вернуть несколько значений из функции Python?

В Python можно вернуть несколько значений из функции, используя кортеж. Для этого возвращаемые значения перечисляются через запятую внутри круглых скобок. В выбираемые значения можно обратиться по индексу.

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

В языке программирования Python значения могут быть переданы в функцию несколькими способами:

- позиционные аргументы (передача аргументов в порядке их следования);
- именованные аргументы (передача аргументов с указанием их имени);
- аргументы по умолчанию (передача аргументов со значениями по умолчанию);
- распаковывание списков и словарей.

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

Для того, чтобы задать значение аргументов функции по умолчанию в Python, нужно указать это значение после имени аргумента в определении функции.

#### 7) Каково назначение lambda-выражений в языке Python?

Lambda-выражения в языке Python представляют собой способ создания анонимных функций без явного определения имени функции. Они могут использоваться как аргументы встроенных функций, таких как `filter()`, `map()` и `reduce()`.

В lambda-выражениях объединяются три элемента: аргументы, оператор-разделитель и тело функции.

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

Документирование кода в Python оформляется с помощью docstring'ов (строки документации), которые помещаются сразу после объявления функции, класса, метода или модуля. Для того чтобы оформить документацию в соответствии с PEP257, используют следующие рекомендации:

1. Docstring должен начинаться с однострочного описания того, что делает объект (функция, класс и т.д.). Это описание следует начинать с заглавной буквы и заканчивать точкой.

2. За однострочным описанием должна следовать пустая строка.

3. Если это функция или метод, то следует указать, какие аргументы она принимает, какие они должны быть по типу и за что они отвечают.

4. Если функция или метод что-то возвращает, то это также должно быть указано в документации.

5. Если объект имеет какие-то особенности или ограничения, то их нужно описать.

6. Если есть примеры использования, то их нужно привести.

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

Однострочная форма документации заключается в одном ряду и применяется для краткого описания. Она начинается со знака # и двух пробелов, специально размещенных после знака #. Многострочная форма документации позволяет вставлять более детальные описания. Она начинается и заканчивается тройными кавычками и предоставляет возможность размещения внутри описания более одного абзаца. Эта форма чаще применяется при описании функций и модулей.

*Вывод:* приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.