

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 №11
дисциплины «Основы программной инженерии»

Выполнила:
Ламская Ксения Вячеславовна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Лабораторная работа 2.8 Работа с функциями в языке Python.

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

Порядок выполнения работы


1. Создание репозитория GitHub.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

 ksenia-lamskaya ▾

Repository name *

/ 11laba

✓ 11laba is available.

Great repository names are short and memorable. Need inspiration? How about **redesigned-funicular** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).



You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создания репозитория

2. Проработала примеры из лабораторной работы.

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

import sys

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(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "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)
        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

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)

    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список 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 2.1 – Код из примера 1

```

>>> add
Фамилия и инициалы? Lamskaya KV
Должность? ghj
Год поступления? 2004
>>> add
Фамилия и инициалы? Svetlakova SA
Должность? gh1
Год поступления? 1877
>>> select 6
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Lamskaya KV              | ghj                  | 2004          |
|  2 | Svetlakova SA            | gh1                  | 1877          |
+-----+-----+-----+-----+

```

Рисунок 2.2 – Вывод программы из примера 1

3. Решить следующую задачу: 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".

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

import sys

def test():
    number = int(input("Введите целое число: "))

    if number > 0:
        positive()
    elif number < 0:
        negative()

def positive():
    print("Положительное")

def negative():
    print("Отрицательное")

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

Рисунок 3.1 – Код программы

```
Введите целое число: 12
Положительное
PS C:\Ksen\11laba\tasks> & C
Ksen/11laba/tasks/1.py
Введите целое число: -6
Отрицательное
```

Рисунок 3.2 – Вывод программы

4. 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 πr^2 . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле

$2\pi rh$, 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():

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

    r = float(input("Введите радиус цилиндра: "))
    h = float(input("Введите высоту цилиндра: "))
    side_area = 2 * math.pi * r * h
    full_area = side_area + 2 * circle(r)

    choice = input("Хотите получить только площадь боковой поверхности цилиндра? ")
    if choice.lower() == "да":
        print(f"Площадь боковой поверхности цилиндра: {side_area}")
    else:
        print(f"Полная площадь цилиндра: {full_area}")

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

Рисунок 4.1 – Код программы

```
Ksen/111laba/tasks/2.py
Введите радиус цилиндра: 15
Введите высоту цилиндра: 10
Хотите получить только площадь боковой поверхности цилиндра? да
Площадь боковой поверхности цилиндра: 942.4777960769379
```

Рисунок 4.2 – Вывод программы

5. 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 -*-

import sys

def check():
    s = 1

    while True:
        num = int(input("Введите число: "))
        if num == 0:
            print(f'Результат: {s}')
            break
        s *= num

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

Рисунок 5.1 – Код программы

```
Введите число: 1
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 5
Введите число: 0
Результат: 120
```

Рисунок 5.2 – Вывод программы

6. Решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2. 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`.

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

4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

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

def get_input():
    n = input("Введите значение: ")

    return n

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

def str_to_int(n):
    s = int(n)
    return s

def print_int(s):
    print(s)

if __name__ == '__main__':
    n = get_input()

    if test_input(n):
        s = str_to_int(n)
        print_int(s)
    else:
        print("Введенное значение не является числом.")
```

Рисунок 6.1 – Код программы

Введите значение: g
Введенное значение не является числом.

Рисунок 6.2 – Вывод программы

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

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

import sys

def help1():
    """
    Функция для вывода списка команд
    """
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

def add1():
    """
    Функция для добавления информации о новых маршрутах
    """
    # Запросить данные о маршруте.
    name = input("Название начального пункта маршрута: ")
    name2 = input("Название конечного пункта маршрута: ")
    number = int(input("Номер маршрута: "))

    # Создать словарь.
```

```

        i = {'name': name, 'name2': name2, 'number': number}

        return i

def error1():
    """
    функция для неопознанных команд
    """
    print(f"Неизвестная команда {command}")

def list(point):
    """
    Функция для вывода списка добавленных маршрутов
    """
    # Заголовок таблицы.
    line = '+--{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(

```

```

        "№",
        "Начальный пункт.",
        "Конечный пункт",
        "№ маршрута"
    )
    )
    print(line)

    # Вывести данные о всех маршрутах.
    for idx, i in enumerate(point, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                idx,
                i.get('name', ''),
                i.get('name2', ''),
                i.get('number', '')
            )
        )
    print(line)

def select(command, point):
    """
    Функция для получения маршрута по его номеру
    """
    # Разбить команду на части для выделения номера маршрута.

```

```

parts = input("Введите значение: ")
# Проверить сведения работников из списка.

for i in point:
    # Проверить сведения.
    flag = True
    for i in point:
        if i['number'] == int(parts):
            print("Начальный пункт маршрута - ", i["name"])
            print("Конечный пункт маршрута - ", i["name2"])
            flag = False
    if flag:
        print("Маршрут с таким номером не найден")

def main():
    """
    Главная функция программы.
    """
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    # Список маршрутов.
    point = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.

        match command:
            case 'exit':
                break

            case 'add':
                # Добавить словарь в список.
                i = add1()
                point.append(i)
                # Отсортировать список в случае необходимости.
                if len(point) > 1:
                    point.sort(key=lambda item: item.get('number', ''))

            case 'list':
                list(point)

            case 'select':
                select(command, point)

            case 'help':
                help1()

            case _:
                error1()

if __name__ == '__main__':
    main()

```

Рисунок 7.1 – Код программы

```

add - добавить маршрут;
list - вывести список маршрутов;
select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Название начального пункта маршрута: a
Название конечного пункта маршрута: b
Номер маршрута: 1
>>> add
Название начального пункта маршрута: v
Название конечного пункта маршрута: b
Номер маршрута: 3
>>> list
+-----+-----+-----+-----+
| № | Начальный пункт. | Конечный пункт | № маршрута |
+-----+-----+-----+-----+
| 1 | a | b | 1 |
| 2 | v | b | 3 |
+-----+-----+-----+-----+
>>> select
Введите значение: 3
Начальный пункт маршрута - v
Конечный пункт маршрута - b

```

Рисунок 7.2 – Вывод программы

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. Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.

2. Каково назначение операторов def и return?

Оператор def в Python используется для определения функции. Он начинает заголовок функции и может принимать ноль или более аргументов, которые могут использоваться в теле функции. Оператор return используется для возврата результата выполнения функции. Он может быть необязательным, так как функция может ничего не возвращать. Оператор return не только возвращает значение, но и производит выход из функции. Поэтому он должен определяться после остальных инструкций. Если функция

не возвращает никакого значения, после оператора `return` не ставится никакого возвращаемого значения.

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

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`.

5. Какие существуют способы передачи значений в функцию?

```
figure4 = cylinder(r=2, h=10)
```

```
def cylinder(h, r=1):
```

```
figure4 = cylinder(i, h)(передаются значения глобальных переменных)
```

6. Как задать значение аргументов функции по умолчанию?

```
def cylinder(h, r=1):
```

7. Каково назначение lambda-выражений в языке Python?

Lambda-выражения в Python, также известные как “анонимные функции”, используются для создания небольших функций без необходимости использования ключевого слова `def`. Они представляют собой компактный способ определения функции.

8. Как осуществляется документирование кода согласно PEP257?

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 используйте `"""triple double quotes"""` для строк документации. 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

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

Однострочные строки документации используются для краткого описания функции, метода, класса или модуля, что делает и какие аргументы принимает. Они заключаются в тройные кавычки и пишутся в императивной форме. Многострочные строки документации используются для более подробного описания функции, метода, класса или модуля, включая их параметры, типы, возвращаемые значения, исключения, примеры и другие детали. Они также заключаются в тройные кавычки, но имеют определенный формат и стиль, в зависимости от выбранной конвенции.

