МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №3
Работа с функциями в языке Python.

по дисциплине «Технологии программирования и алгоритмизации»

Выполнила студентка группы ИВТ-б-о-20-1

Новикова В.С. « »_____20__г.

Подпись студента _____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2021

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

Ход работы:

Ссылка на репозиторий: https://github.com/Valentina1502/LABA_3

Пример 1. (рис. 1).

Код:

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

import sys
from datetime import date


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)


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)

    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


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

```
C:\Users\Valentina\AppData\Local\Programs\Python\Python38\python.exe C:/Users/V
>>> add
Фамилия и инициалы? Иванов И И
Должность? студент
Год поступления? 2020
>>> add
Фамилия и инициалы? Петров О Т
Должность? студент
Год поступления? 2019
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |            Ф.И.О.                |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Иванов И И                       | студент              |     2020 |
|    2 | Петров О Т                       | студент              |     2019 |
+------+----------------------------------+----------------------+----------+
>>> select 2
+------+----------------------------------+----------------------+----------+
|  №   |            Ф.И.О.                |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Петров О Т                       | студент              |     2019 |
+------+----------------------------------+----------------------+----------+
```

Рисунок 1 – Пример 1

Задача 1 (рис. 2):

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".

Код:

```python
#!/urs/bin/env python3
# -*- coding^utf-8 -*-


def test():
    ch = int(input("Введите число: "))
    if ch < 0:
        negative(ch)
    elif ch > 0:
        positive(ch)
    else:
        print(f"Число {ch} = 0")


def positive(ch):
    print(f"Число положительное ({ch})")


def negative(ch):
    print(f"Число отрицательное ({ch})")


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

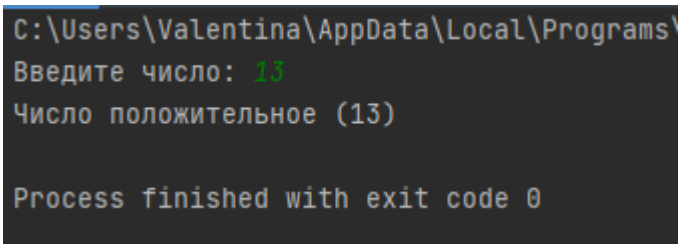

Рисунок 2 – Задача 1

Задача 2 (рис. 3):

В основной ветке программы вызывается функция cylinder(), которая вычисляет площадь цилиндра. В теле cylinder() определена функция circle(), вычисляющая площадь круга по формуле . В теле cylinder() у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции circle().

Код:

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


def circle(rad):
    p_circle = rad * rad * 3.14
    return p_circle


def cylinder():
    radius = float(input("Радиус цилиндра: "))
    height = float(input("Высота цилиндра: "))

    mes = input("Для вывода площади боковой поверхности"
                " введиите 1\nДля вывода полной площади"
```

```
                " цилиндра введите 2\n"
                " >>>> ")
    if mes.lower() == '1':
        bok = 2 * 3.14 * height * radius
        print(f'Площадь боковой поверхности = {bok}')
    elif mes.lower() == '2':
        all_p = 2 * 3.14 * height * radius * (2 * circle(radius))
        print(f'Площадь всей поверхности = {all_p}')
    else:
        print("Команда не опознана")


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

```
C:\Users\Valentina\AppData\Local\Programs\Python\
Радиус цилиндра: 10
Высота цилиндра: 1
Для вывода площади боковой поверхности введиите 1
Для вывода полной площади цилиндра введите 2
 >>>> 2
Площадь всей поверхности = 39438.4

Process finished with exit code 0
```

Рисунок 3 – Задача 2

Задача 3 (рис. 4)

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.

Код:

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

def f():
    proizv = 1
    while True:
        mnogitel = int(input("Введите множитель: "))
        proizv *= mnogitel
        if mnogitel == 0:
            print("Произведение = 0")
            break
        else:
            print(f"Произведение = {proizv}")


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

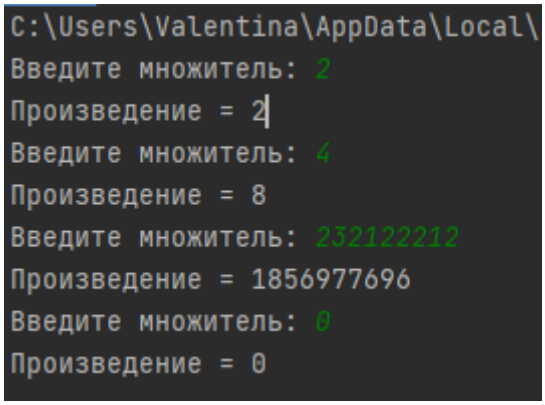

Рисунок 4 – Задача 3

Задача 4 (рис. 5)

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.

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.

Код:

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

def test_input(mes):
    try:
        str_to_int(mes)
    except ValueError:
        print("Строку не преобразовать в число")


def str_to_int(mes):
    i = int(mes)
    print_int(i)


def print_int(i):
    print(i)


def get_input():
    mes = input("Введите строку: ")
    test_input(mes)
```

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

```
C:\Users\Valentina\AppData\Local\Pro
Введите строку: 12
12

Process finished with exit code 0
```

Рисунок 5 – Задача 4

Вариант 11

Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции (рис. 6).

Код:

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

from datetime import date
import sys


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 соответствии с командой
        if command == 'exit':
            break

        elif command == 'add':
            # запрос данных пользователя
            worker = get_worker()
            # добавление словаря в список
            workers.append(worker)
            # сортировка списка в случае необходимости
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('year', ''))

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

        elif command.startswith('found '):

            # разобрать команду на части для выделения номера
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый номер
            phone = (parts[1])

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

```
            display_workers(selected)

        elif command == 'help':
            # Вывести справку о работе с программой
            print("Список команд:\n")
            print("add - добавить работника;")
            print("all - вывести список работников;")
            print("found <x-xxx-xxx-xx-xx> - найти работника по номеру;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

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


def get_worker():
    """
    Запросить данные о работнике.
    """
    # запрос данных пользователя
    name = input("Имя: ")
    fam = input("Фамилия: ")
    year = input("Дата рождения (yyyy.mm.dd): ")
    tel = input("Телефон: (x-xxx-xxx-xx-xx): ")

    # создать словарь
    return {
        'name': name,
        'fam': fam,
        'year': year,
        'tel': tel,
    }


def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 20,
            '-' * 20,
            '-' * 12,
            '-' * 20
        )
        print(line)
        print(
            '| {:^4} | {:^20} | {:^20} | {:^12} | {:^20} |'.format(
                "№",
                "Фамилия",
                "Имя",
                "Год",
                "Телефон"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
```

```
                '| {:^4} | {:^20} | {:^20} | {:^12} | {:^20} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('fam', ''),
                    worker.get('year', ''),
                    worker.get('tel', '')
                )
            )

        print(line)
    else:
        print("Список работников пуст.")


def select_workers(staff, phone):
    """
    Выбрать работников с заданным телефоном.
    """
    result = []
    for employee in staff:
        if employee.get('tel', '') == phone:
            result.append(employee)
    # возвратить список выбранных работников
    return result


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

```
C:\Users\Valentina\AppData\Local\Programs\Python\Python38\python.exe C:/Users/Valentina/Docume
>>>add
Имя: Ваня
Фамилия: Иванов
Дата рождения (yyyy.mm.dd): 2001.04.09
Телефон: (x-xxx-xxx-xx-xx): 8-988-765-43-21
>>>add
Имя: Коля
Фамилия: Петров
Дата рождения (yyyy.mm.dd): 2000.10.23
Телефон: (x-xxx-xxx-xx-xx): 8-943-123-45-65
>>>all
+------+----------------------+----------------------+--------------+----------------------+
|  №   |       Фамилия        |         Имя          |     Год      |       Телефон        |
+------+----------------------+----------------------+--------------+----------------------+
|  1   |         Коля         |        Петров        |  2000.10.23  |   8-943-123-45-65    |
|  2   |         Ваня         |        Иванов        |  2001.04.09  |   8-988-765-43-21    |
+------+----------------------+----------------------+--------------+----------------------+
>>>found 8-943-123-45-65
+------+----------------------+----------------------+--------------+----------------------+
|  №   |       Фамилия        |         Имя          |     Год      |       Телефон        |
+------+----------------------+----------------------+--------------+----------------------+
|  1   |         Коля         |        Петров        |  2000.10.23  |   8-943-123-45-65    |
+------+----------------------+----------------------+--------------+----------------------+
>>>exit
```

Рисунок 6 – Индивидуальное задание

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.

2. Каково назначение операторов def и return ?

Ключевое слово def сообщает интерпретатору, что перед ним определение функции. За def следует имя функции.

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?

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. (return side, full)

5. Какие существуют способы передачи значений в функцию?

Количество аргументов и параметров совпадает. Нельзя передать три аргумента, если функция принимает только два. Нельзя передать один аргумент, если функция требует два обязательных.

Однако 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.

```
def cylinder(h, r=1):
…
return full

figure1 = cylinder(4, 3)
figure2 = cylinder(5)
```

6. Как задать значение аргументов функции по умолчанию?

Явно указать значения для параметров при описании функции

```
def cylinder(h=13, 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.

Небольшой пример:

```
def func(x, y):
return x**2 + y**2

func = lambda x, y: x**2 + y**2
```

Так чем же отличаются так принципиально def и lambda : 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

8. Как осуществляется документирование кода согласно PEP257?

PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код.

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 используйте """triple double quotes""" для строк документации. Используйте r"""raw triple double quotes""" , если вы будете использовать обратную косую черту 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 умещаться на одной строке.

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: при выполнении заданий были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python.