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 ПРАКТИЧЕСКОЙ РАБОТЕ №2.8
дисциплины «Программирование на Python»
Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Работа с функциями в языке Python

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

Ход работы

Пример 1. *Для примера 1 лабораторной работы 2.6, оформить каждую команду в виде вызова отдельной функции.*

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


import sys
from datetime import date


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


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()
```

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Журавлёв А.К
Должность? Директор
Год поступления? 2003
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Журавлёв А.К                     | Директор             |     2003 |
+------+----------------------------------+----------------------+----------+
>>> select 35
Список работников пуст.
>>> select 15
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Журавлёв А.К                     | Директор             |     2003 |
+------+----------------------------------+----------------------+----------+
>>> exit
```

Рисунок 1. Результат работы программы

8. 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".

   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.

В Python порядок определения функций не имеет значения, так как интерпретатор сначала проходит по всему коду и создает объекты для всех функций, а затем начинает выполнение программы. Поэтому определения функций могут следовать после вызова функции test() без каких-либо проблем.

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

def test():
    num = int(input('Введите целое число: '))

    if num > 0:
        positive()
    elif num < 0:
        negative()
    else:
        print('Число 0')


def positive():
    print('Положительное число')


def negative():
    print('Отрицательное число')


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

```
Введите целое число: -7
Отрицательное число

Process finished with exit code 0
```

Рисунок 2. Результат работы программы

10. 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()*.

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


import math


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

    radius = float(input('Радиус цилиндра: '))
    height = float(input('Высота цилиндра: '))

    side_area = 2*math.pi*radius*height

    answer = input('Посчитать полную площадь цилиндра? (y/n) ')
    if answer == 'y':
        full_area = side_area + 2*circle(radius)
        print(f'Полная площадь: {full_area:.2f}')
    elif answer == 'n':
        print(f'Площадь боковой поверхности цилиндра: {side_area:.2f}')
    else:
        print('Неверный ввод')


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

```
Радиус цилиндра: 2
Высота цилиндра: 4
Посчитать полную площадь цилиндра? (y/n) y
Полная площадь: 75.40

Радиус цилиндра: 12
Высота цилиндра: 5
Посчитать полную площадь цилиндра? (y/n) n
Площадь боковой поверхности цилиндра: 376.99
```

Рисунок 3. Результат работы программы

12. 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.

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

def multiplication():
    res = 1
    while True:
        num = int(input('Введите число: '))
        if num == 0:
            break
        res *= num

    return res


if __name__ == "__main__":
    mult = multiplication()
    print('Произведение чисел: ', mult)
```

```
Введите число: 4
Введите число: 3
Введите число: 2
Введите число: 2
Введите число: 0
Произведение чисел:  48
```

Рисунок 4. Результат работы программы

14. 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.

    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.

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


def get_input():
    return input('Введите строку: ')


def test_input(st):
    return st.isdigit() or st[0] == '-'
```

```
def str_to_int(st):
    return int(st)


def print_int(value):
    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('Невозможно преобразовать в целое число')
```

```
Введите строку: -12
Целое число:  -12

Введите строку: 56,7
Невозможно преобразовать в целое число
```

Рисунок 4. Результат работы программы

Индивидуальное задание:

2. Использовать словарь, содержащий следующие ключи: фамилия и инициалы; номер группы; успеваемость (список из пяти элементов). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию среднего балла; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.

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


import sys


def get_student():
    """
    Запросить данные о студенте.
    """
    surname = input("Введите фамилию и инициалы: ")
    group_num = input("Введите номер группы: ")
    print('Введите оценки: ')
    grades = [int(n) for n in input().split()]

    student = {
        'name': surname,
        'group_number': group_num,
        'grades': grades
    }

    return student


def display_students(staff):
    """
```

```
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades'))
                )
            )
        print(line)

    else:
        print("Список студентов пуст.")


def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(grade >= 4 for grade in student['grades']):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result


def main():
    """
    Главная функция программы.
    """
    # Список студентов.
    students = []

    # 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.
            student = get_student()

            # Добавить словарь в список.
            students.append(student)
            # Сортировка по возрастанию среднего балла
            students.sort(key=lambda x: sum(x['grades']) / 5)

        elif command == 'list':
            # Отобразить всех студентов.
            display_students(students)

        elif command == 'select':
            selected = select_students(students)
            # Отобразить выбранных студентов.
            display_students(selected)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить студента;")
            print("list - вывести список студентов;")
            print("select - запросить студентов, имеющих оценки 4, 5;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

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


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

```
>>> help
Список команд:

add - добавить студента;
list - вывести список студентов;
select - запросить студентов, имеющих оценки 4, 5;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Введите фамилию и инициалы: Журавлёв А.В
Введите номер группы: 1
Введите оценки:
5 5 5 4 5
>>> add
Введите фамилию и инициалы: Михайлова С.А
Введите номер группы: 2
Введите оценки:
3 3 4 3 4
>>> add
Введите фамилию и инициалы: Третьяков К.Б
Введите номер группы: 3
Введите оценки:
4 4 4 5 4
>>> list
+------+--------------------------------+----------------------+----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | Михайлова С.А                  | 2                    |  3, 3, 4, 3, 4 |
|    2 | Третьяков К.Б                  | 3                    |  4, 4, 4, 5, 4 |
|    3 | Журавлёв А.В                   | 1                    |  5, 5, 5, 4, 5 |
+------+--------------------------------+----------------------+----------------+
>>> select
+------+--------------------------------+----------------------+----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | Третьяков К.Б                  | 3                    |  4, 4, 4, 5, 4 |
|    2 | Журавлёв А.В                   | 1                    |  5, 5, 5, 4, 5 |
+------+--------------------------------+----------------------+----------------+
>>> exit

Process finished with exit code 0
```

Рисунок 5. Результат выполнения программы

Контрольные вопросы:

1. Функции в Python используются для организации кода, повторного использования и упрощения его понимания. Они позволяют разделить программу на более мелкие и понятные блоки.

2. Оператор def используется для определения функции, а оператор return используется для возврата результата выполнения функции.

3. Локальные переменные существуют только внутри функции и не видны за ее пределами, в то время как глобальные переменные доступны из любой части программы.

4. Для возврата нескольких значений из функции можно просто указать их через запятую после оператора return.

5. Значения могут быть переданы в функцию как позиционные аргументы, ключевые аргументы или в виде списка или словаря при помощи *args и **kwargs соответственно.

6. Значения аргументов функции могут быть заданы по умолчанию прямо при определении функции, указав их после знака равенства.

7. Lambda-выражения используются для создания анонимных (безымянных) функций в Python.

8. Документирование кода согласно PEP257 осуществляется путем добавления строк документации (docstring) к определению функции, модуля или класса.

9. Однострочная форма строк документации заключается в тройных кавычках и располагается на одной строке, а многострочная форма начинается и заканчивается также тройными кавычками, но занимает несколько строк.

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