МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 10
Работа с функциями в языке Python

по дисциплине «Технологии программирования и алгоритмизации»

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

Хацукова А.И. « »_____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 3.x.

Ход работы:

1. Изучила теоретическую часть лабораторной работы и выполнила пример:

```
        #!/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(workers):
    """
    Отобразить список работников.
    """
```

```
    # Проверить, что список работников не пуст.
    if workers:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)
        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(workers, 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.
    """
    workers = []
    # Список работников.
    # 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()
```

2. Далее приступила к выполнению первого общего задания:

```
def zd1():
    sentence = int(input('Введите целое число:'))
    if sentence > 0:
        positive(sentence)
    else:
        negative(sentence)


def negative(sentence):
    print(f'Число {sentence} отрицательное')


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


if __name__ == '__main__':
    zd1()
```

Рисунок 1 – Первое задание

```
C:\Users\Аделаида\Documents\Laba2.8\
Введите целое число: 5
Число 5 положительное

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения первого задания

3. Второе общее задание:

```python
def cylinder():
    radius = int(input('Введите радиус цилиндра: '))
    height = int(input('Введите высоту цилиндра: '))

    def circle():
        print('Площадь полной поверхности цилиндра: ',
              2 * 3.14 * radius * height + 2 * 3.14 * radius ** 2)

    message = input('Какую площать вы хотите получить: площадь бокой'
                    ' поверхности или полную площадь цилиндра?\n'
                    'Площадь бокой поверхности команда - неполная\n'
                    'Площадь полной поверхности цилинда команда - полная\n'
                    '>>>')
    if message.lower() == 'неполная':
        print('Площадь боковой повехности: ', 2 * 3.14 * radius * height)
    elif message.lower() == 'полная':
        circle()
    else:
        print('Неизвестная команда')

if __name__ == '__main__':
```

Рисунок 3 – Второе общее задание

```
C:\Users\Аделайда\AppData\Local\Programs\Python\
The first line: вапро
The second line:унгкпцауистйенпкуовлы
{'п', 'а', 'о', 'в'}

Process finished with exit code 0
```

Рисунок 4 – Результат выполнения второго задания

4. Третье общее задание:

```python
ef zd3():
    sum = 1
    while True:
        message = int(input("Введите число: "))
        sum *= message
        if sum == 0:
            print('Произведение равно 0')
            break
        else:
            print(f'Полчившееся прозведение: {sum}')


f __name__ == '__main__':
    zd3()
```

Рисунок 5 – Третье общее задание

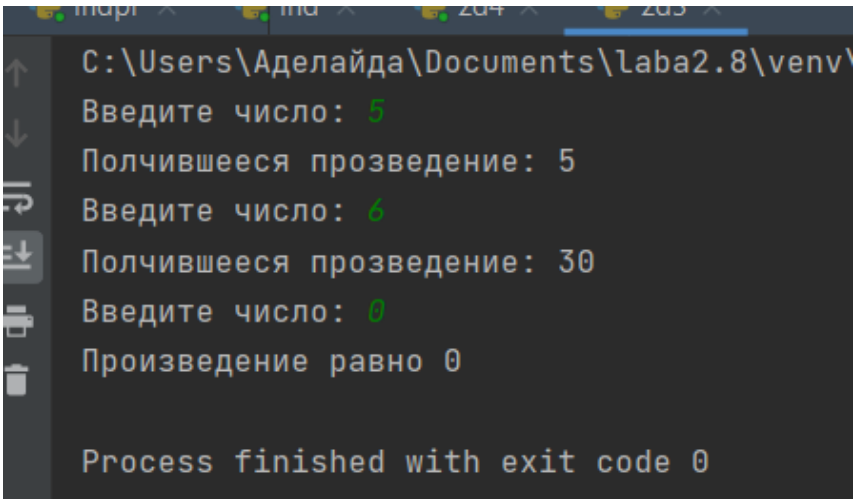

Рисунок 6 – Результат выполнения третьего задания

5. Четвертое общее задание:

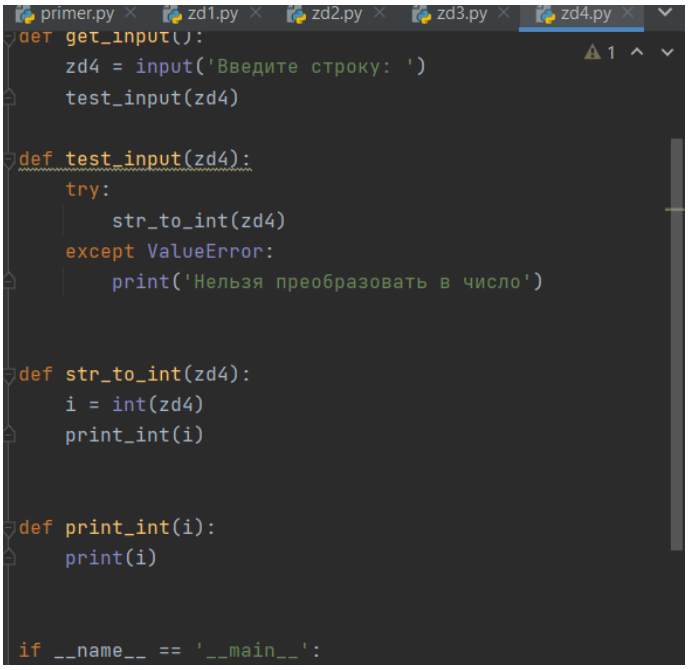

Рисунок 7 – Четвертое общее задание

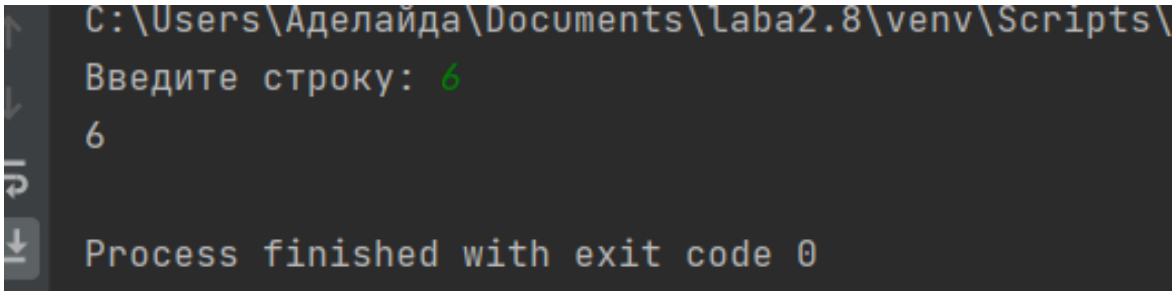

Рисунок 8 – Результат выполнения четвертого примера

6. После приступила к выполнению индивидуального задания по варианту (20 вариант)

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

import sys



def add_student():
    # Запросить данные о студенте.
    name = input("Фамилия и инициалы? ")
    num = input("Номер группы? ")
    print("Успеваемость?  ")
    mat = int(input())
    rus = int(input())
    hist = int(input())
    phis = int(input())
    bio = int(input())
    # Создать словарь.
    student = {
        'name': name,
        'num': num,
        'mat': mat,
        'rus': rus,
        'hist': hist,
        'phis': phis,
        'bio': bio
    }
    # Добавить словарь в список.
    students.append(student)
    # Отсортировать список в случае необходимости.
    if len(students) > 1:
```

```
        students.sort(key=lambda item: item.get('num', ''))



def show_list():
    line = '+-{}-+-{}-+-{}-+-{}-+{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 20,
        '-' * 20,
        '-' * 20,
        '-' * 20,
        '-' * 20
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^20} | {:^20} '
        '| {:^20} | {:^20} | {:^20} |'.format(
            "№",
            "Ф.И.О.",
            "Номер группы",
            "математика",
            "русский",
            "физика",
            "история",
            "биология"
        )
    )
    print(line)
    # Вывести данные о всех студентах.
```

```
    for idx, student in enumerate(students, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>20} | {:>20} '
            '| {:<20} | {:<20} | {:>20} |'.format(
                idx,
                student.get('name', ''),
                student.get('num', ''),
                student.get('mat', ),
                student.get('rus', ),
                student.get('hist', ),
                student.get('phis', ),
                student.get('bio', )
            )
        )
    print(line)


def show_selected():
    # Инициализировать счетчик.
    count = 0
    for student in students:
        averg = (student.get('mat', '') + student.get('rus', '')
                 + student.get('hist', '') + student.get('phis', '')
                 + student.get('bio', '')) / 5
        if averg >= 4:
            count += 1
            print(
                '{:>4}: {}'.format(count, student.get('name', ''))
            )
    # Если счетчик равен 0, то студенты не найдены.
```

```
    if count == 0:
        print("Ученики со средним баллом выше 4-х не найдены.")


def main():
    # 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':
            add_student()

        elif command == 'list':
            show_list()

        elif command.startswith('select'):
            show_selected()

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить студентов;")
            print("list - вывести список студентов;")
            print("select - вывести имена студентов со среднем баллом выше 4")
            print("help - отобразить справку;")
```

```
        print("exit - завершить работу с программой.")

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


if __name__ == '__main__':
    # Список студентов.
    students = []

    main()
```

```
>>> list
+------+--------------------------------+----------------------+----------------------
|  №   |             Ф.И.О.             |     Номер группы     |      математика
+------+--------------------------------+----------------------+----------------------
|    1 | fghdjs                         | 3                    |
+------+--------------------------------+----------------------+----------------------
>>> select
   1: fghdjs
>>>
```

Рисунок 5 – Результат выполнения индивидуального задания

Контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

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, return?

В языке программирования Python функции определяются с помощью оператора def. Ключевое слово def сообщает интерпретатору, что перед ним определение функции. За def следует имя функции. **Оператор return** завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию, в точку, непосредственно следующую за вызовом.

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

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 функций?

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

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. Обратим внимание еще на один момент. Количество аргументов и параметров совпадает. Нельзя передать три аргумента, если функция принимает только два. Нельзя передать один аргумент, если функция требует два обязательных. В рассмотренном примере они обязательные. 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.

6. Как задать значение аргументов функции по умолчанию?

Нужно указать значения для параметров при описании функции

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 PRP257?

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

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

Однострочные 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: в ходе выполнения лабораторной работы были приобретены навыки по работе с функциями на языке программирования Python.