Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5
дисциплины «Программирование на Python»
Вариант №11

Выполнила:

Ковжого Елизавета Андреевна

2 курс, группа ИВТ-б-о-24-1,

09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Проверил:
Воронкин Р. А., доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии.

_____

(подпись)

Отчет защищен с оценкой _____Дата защиты_____

Ставрополь, 2025 г.

Тема: работа с функциями в языке Python.

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

Порядок выполнения работы:

Адес репозитория: https://github.com/LissKovzogo/Python_LAB_6.git

1. Создали, настроили и клонировали репозиторий Python_LAB_6.
2. Создали проект PyCharm в папке репозитория.
3. Проработали все примеры лабораторной работы и создали для каждого отдельный модуль Python.

Листинг кода pr_1.py:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()
        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2
    else:
        return None


if __name__ == '__main__':
    print(median())
    print(median(3, 7, 1, 6, 9))
    print(median(1,5,8,4,3,9))
```

Листинг кода pr_2.py:

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

import sys
from datetime import date

def get_worker():
    name = input("Фамилия и инициалы: ")
    post = input("Должность: ")
    year = int(input("Год поступления: "))
    return {
        "name": name,
        "post": post,
        "year": year,
    }

def display_workers(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} |'.format(
                "№",
                "ФИО",
```

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

        for idx, worker in enumerate(staff,1):
            print(
                '| {:^4} | {:^30} | {:^20} |'.format(
                    idx,
                    worker.get("name",''),
                    worker.get("post",''),
                    worker.get("year",0)
                )
            )
        print(line)

    else:
        print("Список работников пуст.")

def select_worker(staff, period):
    today = date.today()

    result = []
    for employee in staff:
        if today.year - employee.get("year", today.year) >=period:
            result.append(employee)
    return result

def main():
```

```
workers = []

while True:

    command = input(">>> ").lower()

    if command == "exit":
        break

    elif command == "add":
        worker = get_worker()
        workers.append(worker)

        if len(workers) > 1:
            workers.sort(key=lambda item: item.get("name",""))

    elif command == "list":

        display_workers(workers)

    elif command.startswith("select "):

        parts = command.split(" ", maxsplit=1)
        period = int(parts[1])

        selected = select_worker(workers, period)
        display_workers(selected)

    elif command == "help":
        print("Список команд:\n")
```

```
            print("add - добавить работника")
            print("list - вывести список работников")
            print("select <стаж> - запросить работников со стажем")
            print("help - отоброзить справку")
            print("exit - завершить работу с программой")

        else:
            print(f" Неизвестная команда {command}", file=sys.stderr)
            return 1
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

Листинг кода pr_3.py:

```
def recursion(n):
    if n == 1:
        return 1
    return n + recursion(n-1)
print(recursion(5))
```

4. Выполнили индивидуальные задания согласно варианту.

Индивидуальное задание №1: Использовать словарь, содержащий следующие ключи: фамилия и имя, номер телефона, дата рожденияю. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по дате рождения; вывод на экран информации о человеке, номер телефона, которого введён с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

Листинг кода задания №1:

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-

import sys
from datetime import date

def get_contact():
    name = input("Фамилия и инициалы?  ")
    number = input("Номер телефона? ")
    birth_date = datetime.strptime(input("Дата рождения (в формате DD.MM.YYYY)? "), "%d.%m.%Y")

    if number[0] == '+':
        number = '8' + number[2:]
    return {
        'name': name,
        'number': number,
        'date': birth_date,
    }


def display_contacts(contacts):
    if contacts:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 35,
            '-' * 20,
            '-' * 20
        )
        print(line)
        print(
```

```
            '| {:^4} | {:^30} | {:^25} | {:^20} |'.format(
                "№", "ФИО", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        for idx, contact in enumerate(contacts,1):
            date_str = (contact.get('date', '')).strftime("%d.%m.%Y")

            print(
                '| {:>4} | {:<30} | {:<25} | {:<20} |'.format(
                    idx,
                    contact.get('name', ''),
                    contact.get('number', ''),
                    date_str
                )
            )
            print(line)

    else:
        print("Список пуст.")

def select_contact(contacts, num):
    result = []
    for contact in contacts:
        if contact.get('number') == num:
            result.append(contact)
    return result

def main():
```

```
contacts = []

while True:

    command = input(">>> ").lower()

    if command == "exit":
        break

    elif command == "add":
        contact = get_contact()
        contacts.append(contact)

         if len(contacts) > 1:
            contacts.sort(key=lambda item: item.get('date',''))
    elif command == "list":

        display_contacts(contacts)

    elif command.startswith("select "):

        parts = command.split(" ", maxsplit=1)
        num = parts[1]

        selected = select_contact(contacts, num)
        display_contacts(selected)

    elif command == "help":
        print("Список команд:\n")
        print("add - добавить номер")
```

```
            print("list - вывести список контактов")
            print("select <номер> - запросить контакт по номеру")
            print("help - отоброзить справку")
            print("exit - завершить работу с программой")

        else:
            print(f" Неизвестная команда {command}", file=sys.stderr)
            return 1
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

Задание 2: Создайте функцию log_event(event_type, *messages, **options), которая выводит сообщения лога в формате: [LEVEL] (SOURCE) - MESSAGE. Параметры **options могут содержать: level - уровень лога (по умолчанию "INFO"); source - источник (по умолчанию "SYSTEM").

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def log_event(event_type, *messages, **options):

    level = options.get("level", "INFO")
    source = options.get("source", "SYSTEM")

    message = " ".join(str(msg) for msg in messages)

    print(f"[{level}] ({source}) – {message}")

if __name__ == '__main__':
```

log_event("start", "Initialization complete", level="DEBUG", source="Core")

log_event("shutdown", "System shutting down", level="WARNING")

log_event("test", "Message 1", "Message 2", "Message 3", level = "key", source = "value")

Задание 3: Создайте функцию max_depth(data), которая определяет максимальную глубину вложенности списка.

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

def max_depth(a, n = 1):
    depth = n
    for i in a:
        if type(i) is list:
            n += 1
            depth = max(depth, max_depth(i, n))
    return depth

if __name__ == '__main__':
    print(max_depth([1,[2,[3,[4]]]]))
    print(max_depth([1, 2, [3, 4]]))
```

5. Зафиксировали все изменения в репозиторий.

Контрольные вопросы:

1.Каково назначение функций в языке программирования Python? Функции служат для организации многократно используемого кода, разбиения программы на логические блоки, избежания повторения и упрощения чтения кода.

2.Каково назначение операторов def и return? def используется для объявления функции, return — для возврата значения из функции и завершения ее работы.

3.Каково назначение локальных и глобальных переменных при написании функций в Python? Локальные переменные существуют только внутри функции и недоступны извне. Глобальные переменные определены вне функций и доступны во всей программе.

4.Как вернуть несколько значений из функции Python? Несколько значений возвращаются через запятую после return. Фактически возвращается кортеж, который можно распаковать. Пример:

```
def example():
return 1, 2, 3
a, b, c = example()
```

5.Какие существуют способы передачи значений в функцию? Позиционные аргументы (передаются по порядку) и именованные аргументы (передаются по имени параметра).

6.Как задать значение аргументов функции по умолчанию?

Значение по умолчанию задается при объявлении функции через знак равенства. Пример:

```
def greet(name="мир"):
print(f"Привет, {name}")
```

7.Каково назначение lambda-выражений в языке Python?

Lambda-выражения создают анонимные функции для простых операций, часто используются как аргументы в функциях map, filter, sorted.

8.Как осуществляется документирование кода согласно PEP257?

Документирование выполняется с помощью строк документации (docstring), которые размещаются в тройных кавычках сразу после объявления функции, класса или модуря.

9.В чем особенность однострочных и многострочных форм строк документации? Однострочная строка — краткое описание в одну строку.

Многострочная — сначала краткое описание, затем пустая строка, затем подробное описание.

10. Какие аргументы называются позиционными в Python? Позиционные аргументы передаются в функцию в порядке, указанном при ее объявлении.

11. Какие аргументы называются именованными в Python?

Именованные аргументы передаются с указанием имени параметра, что позволяет менять их порядок.

12. Для чего используется оператор *?

Оператор * используется для распаковки итерируемых объектов в аргументы функции и для захвата переменного числа позиционных аргументов.

13. Каково назначение конструкций *args и **kwargs?

*args захватывает произвольное число позиционных аргументов в кортеж.

**kwargs захватывает произвольное число именованных аргументов в словарь.

14. Для чего нужна рекурсия?

Рекурсия нужна для решения задач, которые можно разбить на более простые подзадачи того же типа, например, обход деревьев, вычисление факториала.

15. Что называется базой рекурсии?

Базой рекурсии называется условие выхода из рекурсии, которое останавливает рекурсивные вызовы и предотвращает зацикливание.

16. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций?

Стек программы — это область памяти, которая хранит информацию о вызовах функций. При вызове функции в стек помещается ее контекст (аргументы, локальные переменные, адрес возврата). При завершении функции контекст извлекается из стека.

17. Как получить текущее значение максимальной глубины рекурсии в языке Python? Текущее значение можно получить с помощью функции sys.getrecursionlimit().

18. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python? Возникнет исключение RecursionError, и программа завершится аварийно, если его не обработать.

19. Как изменить максимальную глубину рекурсии в языке Python? Изменить глубину рекурсии можно с помощью функции sys.setrecursionlimit(новый_лимит).

20. Каково назначение декоратора lru_cache? Декоратор lru_cache из модуля functools используется для мемоизации результатов функции, что ускоряет повторные вызовы с теми же аргументами.

21. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов? Хвостовая рекурсия — это рекурсия, где рекурсивный вызов является последней операцией в функции. Стандартный Python не поддерживает оптимизацию хвостовой рекурсии, но ее можно эмулировать или использовать в других языках для предотвращения переполнения стека

Вывод: приобрели навыки по работе с функциями и возвращаемыми значениями при написании программ на языке программирования Python версии 3.x., познакомились с рекурсивными функциями, с областями видимости переменных, с параметрами и аргументами функций, с функциями с произвольным количеством аргументов и с параметрами заданными по умолчанию, lambda-функциями, рекурсивными функциями.