МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.8 по дисциплине основы программной инженерии

Выполнил:
Кожухов Филипп Денисович,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной
безопасности,Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ:

```python
#!/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(
                "No",
                "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)
```

Рисунок 1 - Код программы

```
>>> 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? Dkjgfd D.G.
Должность? Sdaagf
Год поступления? 2001
>>> add
Фамилия и инициалы? Jfkjgl J.K.
Должность? 2012
Год поступления? 2012
>>> list
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Dkjgfd D.G.                      | Sdaagf               |     2001 |
+------+----------------------------------+----------------------+----------+
|    2 | Jfkjgl J.K.                      | 2012                 |     2012 |
+------+----------------------------------+----------------------+----------+
```

Рисунок 2 - Вывод программы

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

def test():
    a = int(input("Enter the number: "))
    if a > 0:
        positive()
    elif a < 0:
        negative()


def positive():
    print("This number is positive")


def negative():
    print("This number is negative")


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

Рисунок 3 - Код задания №1

```
Enter the number: -74
This number is negative

Enter the number: 74
This number is positive
```

Рисунок 4 - Вывод программы

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

import math


def cylinder():

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

    radius = float(input("Enter the radius of the cylinder: "))
    height = float(input("Enter the height if the cylinder: "))
    print("Do you want to compute the full cylinder square? - 'yes' or 'no'")
    command = input().lower()
    if command == 'yes':
        print(2 * math.pi * radius * height + 2 * circle(radius))
    if command == 'no':
        print(2 * math.pi * radius * height)


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

Рисунок 5 - Код задания №2

```
Enter the radius of the cylinder: 12
Enter the height if the cylinder: 5
Do you want to compute the full cylinder square? - 'yes' or 'no'
yes
1281.7698026646356
```

Рисунок 6 - Вывод программы

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

def multiplication():
    mp = 1
    a = int(input("Enter the number: "))
    while a != 0:
        mp *= a
        a = int(input("Enter the number: "))
    return mp


if __name__ == '__main__':
    print(f"The multiplication of entered numbers is: {multiplication()}")
```

Рисунок 7 - Код задания №3

```
Enter the number: 32
Enter the number: 42
Enter the number: 51
Enter the number: 26
Enter the number: 0
The multiplication of entered numbers is: 1782144
```

Рисунок 8 - Вывод программы

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

def get_input():
    a = input("Enter the number: ")
    return a


def test_input(a):
    return a.isdigit()


def str_to_int(a):
    a = int(a)
    return a


def print_int(a):
    print(a)


if __name__ == '__main__':
    num_str = get_input()
    if test_input(num_str):
        num = str_to_int(num_str)
        print_int(num)
    else:
        print(f"The input {num_str} is not numerical")
```

Рисунок 9 - Код задания №4

```
Enter the number: 25
25

Enter the number: io42
The input io42 is not numerical
```

Рисунок 10 - Примеры работы программы

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 5

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

import sys


def get_flight():
    """
    Запросить данные о полёте
    """
    flight_destination = input("Введите название пункта назначения ")
    flight_number = input("Введите номер рейса ")
    airplane_type = input("Введите тип самолета ")
    return {
        'flight_destination': flight_destination,
        'flight_number': flight_number,
        'airplane_type': airplane_type,
    }


def display_flights(flights):
    """
     Отобразить список рейсов
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Пункт назначения",
                "Номер рейса",
                "Тип самолета"
            )
        )
        print(line)

        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
                    idx,
                    flight.get('flight_destination', ''),
                    flight.get('flight_number', ''),
                    flight.get('airplane_type', 0)
                )
            )
        print(line)

    else:
        print("Список рейсов пуст")


def select_flights(flights, airplane_type):
    """
    Выбрать рейсы самолётов заданного типа
    """
    count = 0
    res = []
    for flight in flights:
        if flight.get('airplane_type') == airplane_type:
            count += 1
            res.append(flight)
    if count == 0:
        print("рейсы не найдены")

    return res


def main():
    """
    Главная функция программы
    """
    flights = []
    while True:
        command = input(">>> ").lower()
```

Рисунок 11 - Код индивидуального задания

```
            if command == 'exit':
                break

            elif command == 'add':
                flight = get_flight()
                flights.append(flight)
                if len(flights) > 1:
                    flights.sort(
                        key=lambda item:
                        item.get('flight_destination', ''))

            elif command == 'list':
                display_flights(flights)

            elif command.startswith('select '):
                parts = command.split(' ', maxsplit=1)
                airplane_type = (parts[1].capitalize())
                print(f"Для типа самолета {airplane_type}:")
                selected = select_flights(flights, airplane_type)
                display_flights(selected)

            elif command == 'help':
                # Вывести справку о работе с программой.
                print("Список команд:\n")
                print("add - добавить рейс;")
                print("list - вывести список всех рейсов;")
                print("select <тип самолета> - запросить рейсы указанного типа "
                      "самолета;")
                print("help - отобразить справку;")
                print("exit - завершить работу с программой.")
            else:
                print(f"Неизвестная команда {command}", file=sys.stderr)


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

Рисунок 12 - Код индивидуального задания, продолжение

```
>>> help
Список команд:

add - добавить рейс;
list - вывести список всех рейсов;
select <тип самолета> - запросить рейсы указанного типа самолета;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Введите название пункта назначения Tokyo
Введите номер рейса A2540
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Los Angeles
Введите номер рейса F524P
Введите тип самолета Sanitary
>>> add
Введите название пункта назначения Oslo
Введите номер рейса M5294
Введите тип самолета Military
>>> list
+------+--------------------------------+----------------------+-----------------+
|  No  |        Пункт назначения        |     Номер рейса      |  Тип самолета   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Los Angeles                    | F524P                | Sanitary        |
|    2 | Oslo                           | M5294                | Military        |
|    3 | Tokyo                          | A2540                | Passenger       |
+------+--------------------------------+----------------------+-----------------+
>>>
```

Рисунок 13 - Вывод программы

ОТВЕТЫ НА ВОПРОСЫ

1. 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. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы. 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. 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. Оно может быть любым, так же, как и всякий идентификатор, например, переменная.

3. 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.

4. В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. 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.

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

6. В строке объявления функции указать в скобках значение параметра по умолчанию.

7. lambda функции позволяют определять небольшие однострочные функции на лету. 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. Есть и еще одно интересное применение - хранение списка обработчиков данных в списке/словаре.

8. 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 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. Одиночные строки документации предназначены для действительно очевидных случаев. Они должны умещаться на одной строке. Используйте тройные кавычки, даже если документация умещается на одной строке. Потом будет проще её дополнить. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции). 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. Вставляйте пустую строку до и после всех строк документации (однострочных или многострочных), которые документируют класс - вообще говоря, методы класса разделены друг от друга одной пустой строкой, а строка документации должна быть смещена от первого метода пустой строкой; для симметрии, поставьте пустую строку между заголовком класса и строкой документации. Строки документации функций и методов, как правило, не имеют этого требования.