

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Работа с функциями в языке Python»**

**Отчет по лабораторной работе № 2.8
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова В.Н. « » 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Зафиксируйте изменения в репозитории.

Пример 1. Определить результат выполнения операций над множествами. Считать элементы множества строками.

Код:

```
#!/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()

```

```
add
Фамилия и инициалы: Кипятков А.Н.
Должность: повар
Год поступления: 2009
>>> add
Фамилия и инициалы: Белый С.Ю.
Должность: шеф
Год поступления: 2001
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Белый С.Ю. | шеф | 2001 |
| 2 | Кипятков А.Н. | повар | 2009 |
+-----+-----+-----+-----+
>>>
```

Рисунок 6 – Результат работы программы

8. Решите 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".

Код:

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

def positive():
    print('Positive')

def negative():
    print('Negative')

def test():
    if a > 0:
        positive()
    elif a == 0:
        print("Zero")
```

```
elif a < 0:
    negative()

if __name__ == '__main__':
    a = int(input('Enter chislo: '))

    test()
```

```
test() > elif a == 0
```

1 x

C:\Users\vika1\AppData\Local\Microsoft\WindowsApps

Enter chislo: 34

Positive

Process finished with exit code 0

Рисунок 7 – Результат работы программы

1 x

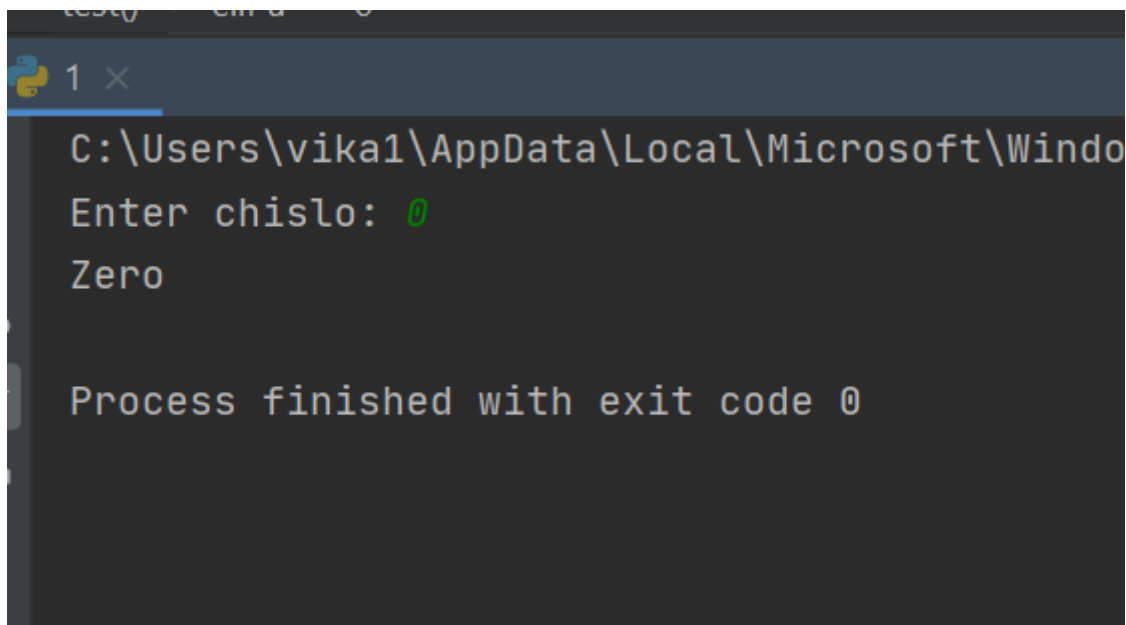
C:\Users\vika1\AppData\Local\Microsoft\WindowsA

Enter chislo: -200

Negative

Process finished with exit code 0

Рисунок 8 – Результат работы программы



```
C:\Users\vika1\AppData\Local\Microsoft\Windows
Enter chislo: 0
Zero

Process finished with exit code 0
```

Рисунок 9 – Результат работы программы

9. Зафиксируйте сделанные изменения в репозитории.

10. Решите задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле $S = \pi r^2$. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле $S_{\text{бок}} = 2\pi r h$, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`

Код:

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

import math
import sys

def cylinder():
    """
    Вычисление площади цилиндра
    """
    r = int(input("Введите радиус: "))
    h = int(input("Введите высоту: "))
    while True:
        command = int(input("Выберете варинат:\n"
                             "1: площадь боковой поверхности цилиндра\n"
                             "2: площадь цилиндра\n"))
```

```

if command == 1:
    s = 2 * math.pi * r * h
    print("S(боковая) = ", s)
    break
elif command == 2:
    s = (2 * math.pi * r * h) + (circle(r) * 2)
    print("S(полная) = ", s)
    break

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

def circle(r):
    """
    Вычисление площади круга в цилиндре
    """
    return math.pi * (r ** 2)

def main():
    """
    Главная функция программы
    """
    cylinder()

if __name__ == '__main__':
    main()

```

```

2 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python
Введите радиус: 4
Введите высоту: 6
Выберите вариант:
1: площадь боковой поверхности цилиндра
2: площадь цилиндра
1
S(боковая) = 150.79644737231007

Process finished with exit code 0

```

Рисунок 12 – Результат работы программы

11. Зафиксируйте сделанные изменения в репозитории.

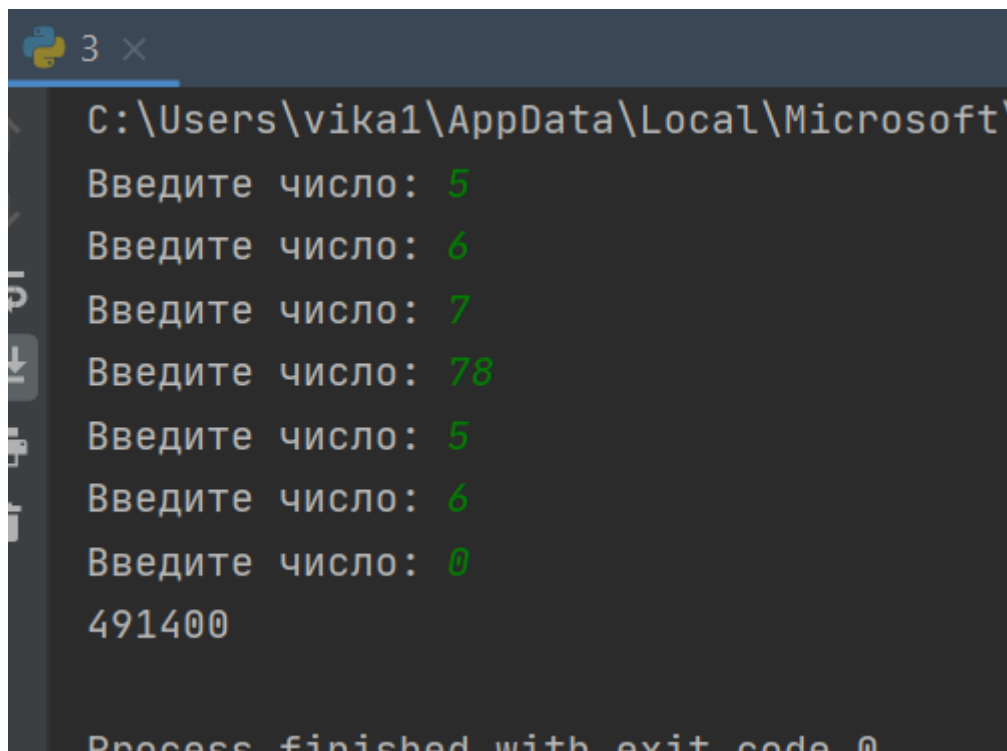
12. Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

Код:

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

def multiplication():
    p = 1
    while True:
        a = int(input('Введите число: '))
        if a == 0:
            break
        else:
            p *= a
    print(p)

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



```
3 x
C:\Users\vika1\AppData\Local\Microsoft\
Введите число: 5
Введите число: 6
Введите число: 7
Введите число: 78
Введите число: 5
Введите число: 6
Введите число: 0
491400
Process finished with exit code 0
```

Рисунок 13 – Результат работы программы

13. Зафиксируйте изменения в репозитории.

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.

Код:

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

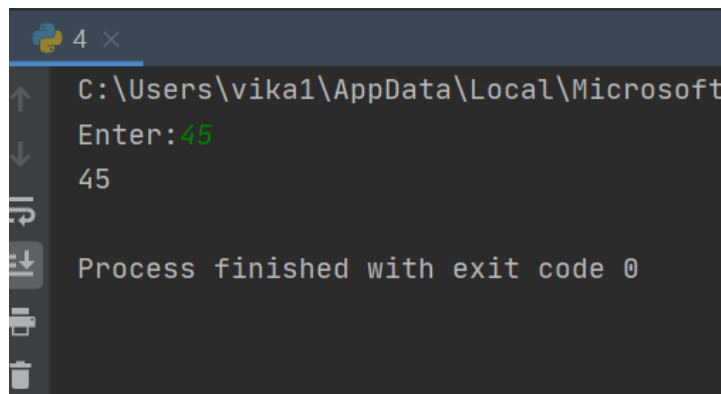
def get_input():
    a = input('Enter:')
    return a

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

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

def print_int(i):
    print(i)

if __name__ == '__main__':
    s = get_input()
    if test_input(s):
        print_int(str_to_int(s))
```



```
4 x
C:\Users\vika1\AppData\Local\Microsoft
Enter: 45
45
Process finished with exit code 0
```

Рисунок 18 – Результат работы программы

15. Зафиксируйте изменения в репозитории.

16. Приведите в отчете скриншоты работы программ решения индивидуального задания.

Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Использовать словарь, содержащий следующие ключи: название пункта назначения;
номер
поезда; время отправления. Написать программу, выполняющую следующие
действия:
ввод с клавиатуры данных в список, состоящий из словарей заданной структуры;
записи должны
быть упорядочены по номерам поездов;
вывод на экран информации о поезде, номер которого введен с клавиатуры; если
таких поездов нет,
выдать на дисплей соответствующее сообщение.
"""

import sys

def add(trains, name, num, time):
    train = {
        'name': name,
        'num': num,
        'time': time,
    }
    trains.append(train)
    if len(trains) > 1:
        trains.sort(key=lambda item: item.get('num', ''))

def listt(trains):
```

```

line = '+-{}-+-{}-+-{}-+-{}-+'.format(
    '-' * 4,
    '-' * 30,
    '-' * 20,
    '-' * 17
)
print(line)
print(
    '| {:^4} | {:^30} | {:^20} | {:^17} |'.format(
        "№",
        "Пункт назначения",
        "Номер поезда",
        "Время отправления"
    )
)
print(line)

for idx, train in enumerate(trains, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>17} |'.format(
            idx,
            train.get('name', ''),
            train.get('num', ''),
            train.get('time', 0)
        )
    )

print(line)

def select(trains, number):
    count = 0
    for train in trains:
        if train.get('num') == number:
            count += 1
            print('Номер поезда:', train.get('num', ''))
            print('Пункт назначения:', train.get('name', ''))
            print('Время отправления:', train.get('time', ''))

    if count == 0:
        print("Таких поездов нет!")

def main():
    trains = []

    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            name = input("Название пункта назначения: ")
            num = int(input("Номер поезда: "))
            time = input("Время отправления: ")

            add(trains, name, num, time)

        elif command == 'list':
            listt(trains)

        elif command.startswith('select'):
            parts = command.split(' ', maxsplit=2)

```

```

        number = int(parts[1])
        select(trains, number)

    elif command == 'help':
        print("Список команд:\n")
        print("add - добавить поезд;")
        print("list - вывести список поездов;")
        print("select <номер поезда> - запросить информацию о выбранном
поезде;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")

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

if __name__ == '__main__':
    main()

```

```

list
+-----+-----+-----+-----+
|  №  | Пункт назначения |  Номер поезда  |  Время отправления  |
+-----+-----+-----+-----+
|   1  | Stavropol        |   2            |          15:30      |
|   2  | Moscow           |   3            |          12:00      |
|   3  | Sochi            |   7            |          19:20      |
+-----+-----+-----+-----+
>>> select 3
Номер поезда: 3
Пункт назначения: Moscow
Время отправления: 12:00
>>>

```

Рисунок 18 – Результат работы программы

17. Зафиксируйте сделанные изменения в репозитории.

Вопросы для защиты работы

1. Каково назначение функций в языке программирования Python?

Функция – это средство (способ) группирования фрагментов программного кода таким образом, что этот программный код может вызываться многократно с помощью использования имени функции. Использование функций в программах на Python даёт следующие взаимосвязанные преимущества: избежание повторения одинаковых фрагментов кода в разных частях программы; уменьшение избыточности исходного кода программы. Как следствие, уменьшение логических ошибок программирования; улучшенное восприятие исходного кода программы в случаях, где вместо блоков многочисленных инструкций (операторов) вызываются имена готовых протестированных функций. Это, в свою очередь, также уменьшает количество ошибок; упрощение внесения изменений в повторяемых блоках кода, организованных в виде функций. Достаточно внести изменения только в тело функции, тогда во всех вызовах данной функции эти изменения будут учтены; с помощью функций удобно разбивать сложную систему на более простые части. Значит, функции – удобный способ структурирования программы; уменьшение трудозатрат на программирование, а, значит, повышение производительности работы программиста.

2. Каково назначение операторов def и return?

Оператор def, выполняемый внутри определения функции, определяет локальную функцию, которая может быть возвращена или передана. Свободные переменные, используемые во вложенной функции, могут обращаться к локальным переменным функции, содержащей def.

Оператор return [выражение] возвращает результат из функции. Оператор return без аргументов аналогичен return None

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

Области видимости определяют, в какой части программы мы можем работать с той или иной переменной, а от каких переменная «скрыта».

Так глобальные переменные доступны в любой точке программы, а локальные переменные, только в функциях, где они объявлены.

4. Как вернуть несколько значений из функции Python?

С помощью оператора `return`. Чтобы вернуть несколько значений, нужно написать их через запятую.

5. Какие существуют способы передачи значений в функцию?

Существует два способа передачи параметров в функцию: по значению и по адресу. При передаче по значению на месте формальных параметров записываются имена фактических параметров. При вычислении функции в стек заносятся копии значений фактических параметров, и операторы функции работают с этими копиями.

6. Как задать значение аргументов функции по умолчанию?

В Python аргументам функции можно присваивать значения по умолчанию, используя оператор присваивания `=`.

7. Каково назначение `lambda`-выражений в языке Python?

Лямбда-выражения на Python - конструкторы простых безымянных однострочных функций. Могут быть использованы везде, где требуется.

8. Как осуществляется документирование кода согласно PEP257?

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 РЕР описывает соглашения, а не правила или синтаксис.

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

Одиночные строки документации предназначены для действительно очевидных случаев.

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.