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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.8 Работа с функциями в языке Python»

ОТЧЕТ
по лабораторной работе №11
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

Лабораторная работа 2.8 Работа с функциями в языке Python

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

Ход работы:

Задание 1

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.

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


def test():
    num = int(input("Enter number: "))
    if num > 0:
        positive()
    elif num < 0:
        negative()
    else:
        print("It's zero")


def positive():
    print("Positive")


def negative():
    print("Negative")


if __name__ == '__main__':
    test()

```

Рисунок 1 – Код программы для задания 1

```
E:\GitHub\laba11\user\Scripts\python.exe E:\GitHub\laba11\test.py
Enter number: 7
Positive
```

Рисунок 2 – Результат работы программы для задания 1

Порядок определения функций positive() и negative() относительно друг друга значения не имеет. Так как выполнение функции зависит от её вызова, а не определения.

Задание 2

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 ...


DFooRS
def cylinder():
    r = float(input("Enter radius: "))
    h = float(input("Enter height: "))

    DFooRS
    def circle():
        return math.pi * r * r

    n = input("Are you wanna get S full or side S? ")

    if n == "full":
        print(circle() * 2 + 2 * math.pi * r * h)
    elif n == "side":
        print(2 * math.pi * r * h)
    else:
        print('Enter "full" or "side"! ', file=sys.stderr)


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

Рисунок 3 – Код программы для задания 2

```
Enter radius: 5
Enter height: 10
Are you wanna get S full or side S? full
471.23889803846896
```

Рисунок 4 – Результат работы программы для задания 2

Задание 3

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 -*-


DFooRS
def multiple():
    global result

    while True:
        n = float(input("Enter number: "))
        if n != 0:
            result *= n
        else:
            return result


if __name__ == '__main__':
    result = 1
    print(multiple())
```

```
multiple
E:\GitHub\laba11\user\Scripts\python.exe E:\GitHub\laba11\multiple.py
Enter number: 4
Enter number: 8
Enter number: 2
Enter number: 0
64.0
```

Рисунок 5 – Код программы и результат работы программы для задания 3

Задание 4

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.

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


DFooRS
def get_input():
    s = input("Enter string: ")
    return s


DFooRS
def test_input(s):
    if s.isdigit():
        return True
    else:
        return False


DFooRS
def str_to_int(s):
    return int(s)


DFooRS
def print_int(s):
    print(s, type(s))
```

Рисунок 6 – Код программы для задания 4

```python
if __name__ == "__main__":
    num = get_input()
    print(num, type(num))

    if test_input(num):
        str_to_int(num)
        print_int(str_to_int(num))
    else:
        print("Conversion not possible")
```

Рисунок 7 – Код программы для задания 4

```
Enter string: 7t
7t <class 'str'>
Conversion not possible
```

Рисунок 8 – Результат работы программы для задания 4

Индивидуальное задание

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

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

import sys


DFooRS
def add_product(products):
    prod = input("Введите название товара: ")
    shop = input("Введите название магазина: ")
    cost = float(input("Введите стоимость товара: "))
    product = {
        'product': prod,
        'shop': shop,
        'cost': cost
    }

    products.append(product)
    if len(products) > 1:
        products.sort(key=lambda item: item.get('shop', ''))
```

Рисунок 9 – Код программы для индивидуального задания

```python
def product_list(products):

    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20
    )
    print(line)
    print(
        '| {:^25} | {:^15} | {:^14} |'.format(
            "Товар",
            "Магазин",
            "Стоимость"
        )
    )
    print(line)

    for product in products:
        print(
            '| {:^25} | {:^15} | {:^14} |'.format(
                product.get('product', ''),
                product.get('shop', ''),
                product.get('cost', 0)
            )
        )
    print(line)
```

Рисунок 10 – Код программы для индивидуального задания

```python
def select(products):
    sel_shop = input("Введите магазин: ")

    n = 0
    for product in products:
        if product.get('shop', '') == sel_shop:
            print(
                '| {:^25} | {:^15} | {:^14} |'.format(
                    product.get('product', ''),
                    product.get('shop', ''),
                    product.get('cost', 0)
                )
            )
            n += 1
    if n == 0:
        print("Магазин не найден!")


 DFooRS
def get_help():
    print("Список команд:\n")
    print("add - добавить информацию о товаре;")
    print("list - вывести список товаров;")
    print("select - запросить товары из одного магазина;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
```

Рисунок 11 – Код программы для индивидуального задания

```python
def error(command):
    print(f"Неизвестная команда {command}", file=sys.stderr)


DFooRS
def main():
    products = []

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

        if command == 'exit':
            break

        elif command == 'add':
            add_product(products)

        elif command == 'list':
            product_list(products)

        elif command == 'select':
            select(products)

        elif command == 'help':
            get_help()

        else:
            error(command)
```

Рисунок 12 – Код программы для индивидуального задания

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

Рисунок 13 - Код программы для индивидуального задания

```
>>> add
Введите название товара: гречка
Введите название магазина: магнит
Введите стоимость товара: 88
>>> add
Введите название товара: макароны
Введите название магазина: пятерочка
Введите стоимость товара: 40
>>> list
+------+--------------------------------+----------------------+
|           Товар           |     Магазин     |   Стоимость    |
+------+--------------------------------+----------------------+
|          гречка           |     магнит      |      88.0      |
|         макароны          |    пятерочка    |      40.0      |
+------+--------------------------------+----------------------+
```

Рисунок 14 – Работа программы для индивидуального задания

Вывод: были приобретены навыки по работе с функциями при написании программ спомощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы:

1. Каково назначение функций в языке программирования Python? 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.

2. Каково назначение операторов def и return? В языке программирования Python функции определяются с помощью оператора def. Оператор return передаёт значение из функции в основную ветку программы. Говорят «функция возвращает значение».

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

4. Как вернуть несколько значений из функции Python? Для этого нужно перечислить эти значения через запятую после оператора return.

5. Какие существуют способы передачи значений в функцию? Через параметры, и через ввод, запрашиваемый самой функцией.

6. Как задать значение аргументов функции по умолчанию? Нужно указать этот параметр после остальных и присвоить ему значение через оператор присваивания «=».

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

8. Как осуществляется документирование кода согласно PEP257? Документирование кода в python - достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. 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? 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.