

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Плещенко Данила Георгиевич
1 курс, группа ИТС-б-о-21-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук, доцент
кафедры инфокоммуникаций

(подпись)

Отчет защищен оценкой _____ Дата защиты _____

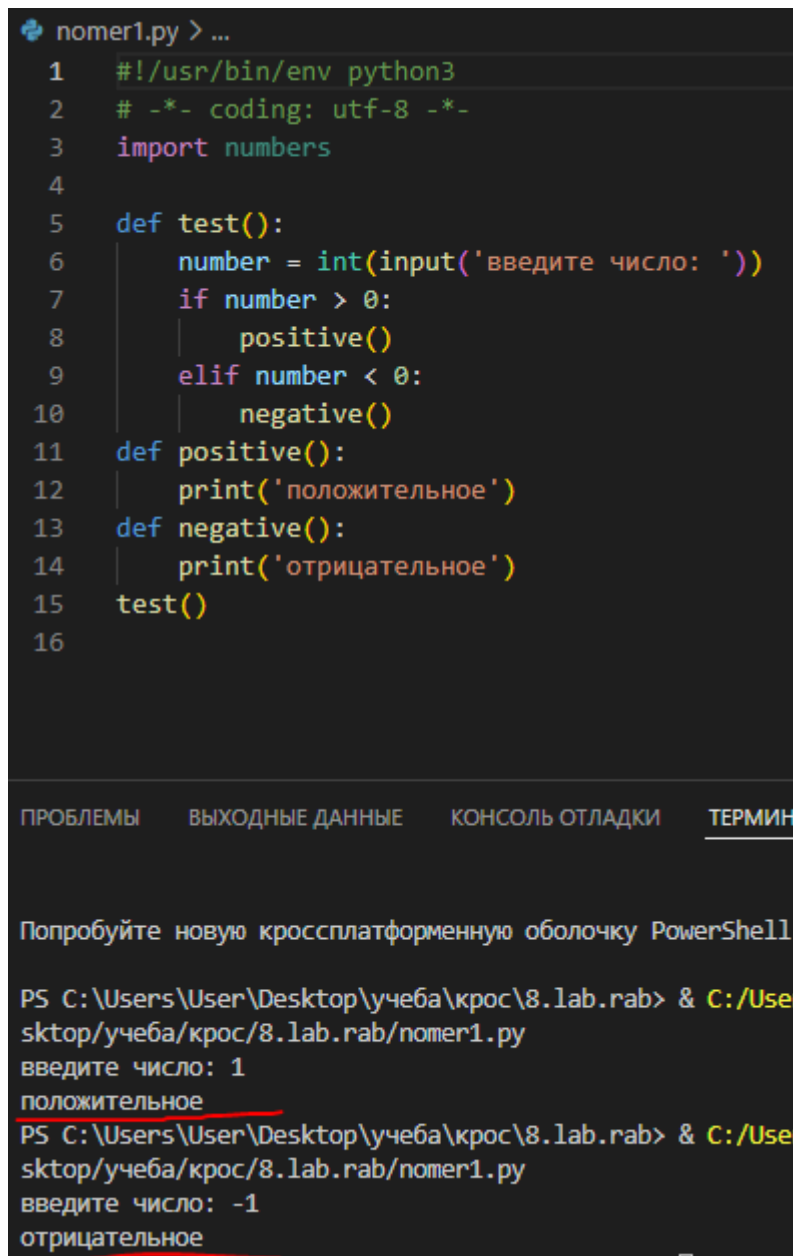
Ставрополь, 2022 г.

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

Ход работы:

Создал новый репозиторий [danilaple/8.lab.rab \(github.com\)](https://github.com/danilaple/8.lab.rab) и начал отработку общих заданий.

Задание 1:



```
nomer1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import numbers
4
5  def test():
6      number = int(input('введите число: '))
7      if number > 0:
8          positive()
9      elif number < 0:
10         negative()
11 def positive():
12     print('положительное')
13 def negative():
14     print('отрицательное')
15 test()
16
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

Попробуйте новую кроссплатформенную оболочку PowerShell

PS C:\Users\User\Desktop\учеба\крос\8.lab.rab> & C:/Users/User/Desktop/учеба/крос/8.lab.rab/nomer1.py
введите число: 1
положительное

PS C:\Users\User\Desktop\учеба\крос\8.lab.rab> & C:/Users/User/Desktop/учеба/крос/8.lab.rab/nomer1.py
введите число: -1
отрицательное

Рисунок 1. Работа программы «Задание 1»

Задание 3:

```
nomer3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  def test():
4      answer = 1
5      while 1:
6          num = int(input())
7          if not num: break
8          answer *= num
9      return(answer)
10 print(test())
11
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТ

```
PS C:\Users\User\Desktop\учеба\крос\8.lab\skriptor/учеба/крос/8.lab.rab/nomer3.py
12
13
1  1
23 23
0  0
3588
```

Рисунок 3. Работа программы «Пример 3»

Вывод: Я приобрёл навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Контрольные вопросы

1. Каково назначение функций в языке программирования Python?

Ответ: 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.

2. Каково назначение операторов def и return ?

Ответ: 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 возвращает значение из функции. return без аргумента возвращает None. Функции, у которых return не определен, также возвращает None.

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

Ответ: В Python переменная, объявленная вне функции или в глобальной области видимости, называется глобальной переменной. К глобальной переменной можно получить доступ как внутри, так и вне функции.

Переменная, объявленная внутри тела функции или в локальной области видимости, называется локальной переменной.

4. Как вернуть несколько значений из функции Python?

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

5. Какие существуют способы передачи значений в функцию?

Ответ: По умолчанию аргументы могут передаваться в функцию Python либо по положению, либо явно по ключевому слову. Для производительности и удобочитаемости имеет смысл ограничить способ передачи аргументов. Где символы / и * являются НЕ обязательными. Эти символы указывают тип аргумента в зависимости от того, как они могут быть переданы в функцию:

только по позиции,

по позиции или по ключевому слову

только по ключевому слову.

6. Как задать значение аргументов функции по умолчанию?

Ответ: Значения параметров по умолчанию создаются при определении функции, а НЕ каждый раз, когда она вызывается в коде программы. Это

означает, что эти выражение вычисляется один раз, и что для каждого вызова используется одно и то же предварительно вычисленное значение. Если функция изменяет объект (например, путем добавления элемента в список, словарь), значение по умолчанию фактически изменяется.

7. Каково назначение lambda-выражений в языке Python?

Ответ:Python поддерживает интересный синтаксис, позволяющий определять небольшиеоднотрочные 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 PEP257?

Ответ: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. При нарушении этих соглашений, самое худшее, чего можно ожидать – некоторыхнеодобрительных взглядов. Но некоторые программы (например, docutils), знают о соглашениях, поэтому следование им даст вам лучшие результаты. Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода.

9. В чем особенность однотрочных и многотрочных форм строк документации?

Ответ:Однотрочные:

```
def kos_root():  
    """Return the pathname of the KOS root directory."""  
    global _kos_root  
    if _kos_root: return _kos_root
```

Используйте тройные кавычки, даже если документация уместается на одной строке. Потом будет проще её дополнить.

Однотрочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции). Не делайте:

```
def function(a, b):  
    """function(a, b) -> list"""
```

Этот тип строк документации подходит только для С функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем интроспекции. Предпочтительный вариант для такой строки документации будет что-то вроде:

```
def function(a, b):  
    """Do X and return a list."""
```

Многострочные:

Многострочные строки документации состоят из однотрочной 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 (см. пример ниже).