

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

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

Кафедра инфокоммуникаций
Основы кроссплатформенного программирования
Отчет по лабораторной работе №8
«Работа с функциями в языке Python»

Выполнил студент группы ИТС-б-о-21-1

Пушкин Максим

« » _____ 2022г.

Подпись студента _____

Проверил: Доцент, к.т.н, доцент кафедры
инфокоммуникаций

Воронкин А. В.

Работа защищена с оценкой: ____

(подпись)

Ставрополь, 2022

Лабораторная работа 3

Лабораторная работа 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:

Решение задач:

Задача 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".

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import numbers
5
6
7  def test():
8      number = int(input('введите число: '))
9      if number > 0:
10         positive()
11     elif number < 0:
12         negative()
13     else:
14         print("я вас не совсем понял. ;)")
15         test()
16 def positive():
17     print('положительное')
18 def negative():
19     print('отрицательное')
20 test()
```



Задача 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()*.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def cylinder(r, h):
5      from math import pi
6      def circle(r): return pi*r**2
7      s = 2*pi*r*h
8      if input('Full area? [y/n]: ') == 'y': \
9          s += 2*circle(r)
10     return (s)
11
12 r, h = 1, 1
13 print('s =', cylinder(r, h))
14
```

Задача 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.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def test():
5      answer = 1
6      while 1:
7          num = int(input())
8          if not num: break
9          answer *= num
10     return(answer)
11 print(test())
12

```

Задача 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.

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.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def getInput(): return(input('запрашиваю ввод: '))
5  def testInput(n):
6      try: n = int(n); return(True)
7      except: return(False)
8  def strToInt(n): return(int(n))
9  def printInt(n): print(n)

```

Индивидуальные задания:

Задание 1.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 from random import randint
5 # создали список из случайных чисел
6 list = [(randint(1, 100)) for _ in range(31)]
7 print(list)
8
9 # i учитывает индекс элемента в списке
10 i = 1
11 # цикл переберет все значения списка начиная с элемента с индексом 1 и до конца (предыдущий элемент имеет индекс = 0)
12 for num in list[1::]:
13     # условие, которое проверяет, соседние элементы
14     if list[i - 1] < num and list[i] < num:
15         print(list[i - 1], list[i], list[i + 1])
16         # при нахождении выходим из цикла
17         break
18     else:
19         # иначе прибавляем к индексу 1 и снова сравниваем
20         i += 1
```

input

7, 3, 78, 62, 60, 39, 25, 19, 96, 87, 55, 61, 43, 19, 9, 77, 46, 28, 59, 7, 30, 60, 15, 86, 78, 35, 34, 20, 58, 73, 90]

78 62

Ответы на 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 это 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 небольшие однострочные 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 – некоторых неодобрительных 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 однострочных и многострочных форм строк документации?

Ответ: Однострочные:

```
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.

Однострочная 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 C 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 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 (см. пример ниже).

Вывод: приобрел навык по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.