

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

Институт цифрового
развития

Кафедра
инфокоммуникаций

ОТЧЕТ

ПОЛАБОРАТОРНОЙ РАБОТЫ №2.8

**Дисциплины «Основы кросс платформенного
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 с оценкой _____ Дата защиты _____

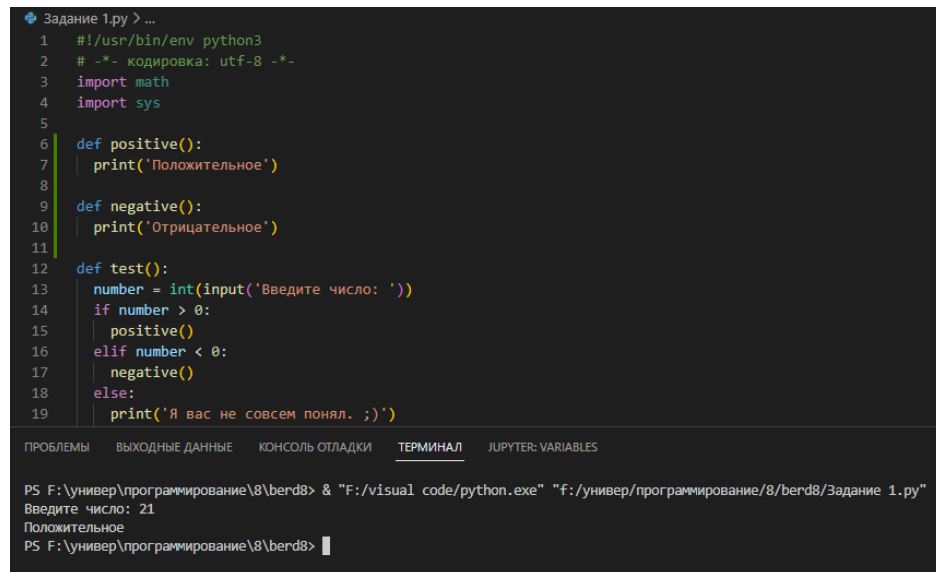
Ставрополь, 2022 г.

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

Ход работы:

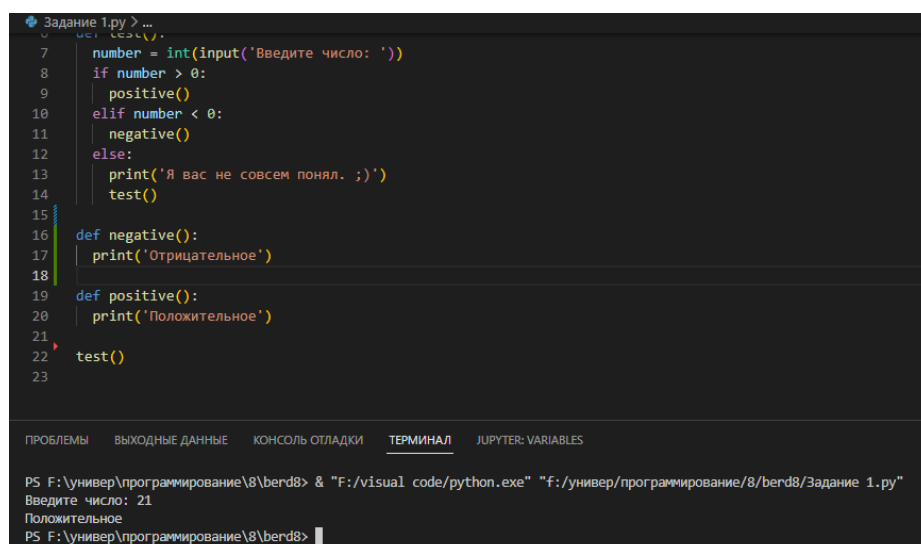
Создал новый репозиторий <https://github.com/Blekroyt/berd8.git>

Задание 1



```
Задание 1.py > ...
1  #!/usr/bin/env python3
2  # -*- кодировка: utf-8 -*-
3  import math
4  import sys
5
6  def positive():
7      print('Положительное')
8
9  def negative():
10     print('Отрицательное')
11
12 def test():
13     number = int(input('Введите число: '))
14     if number > 0:
15         positive()
16     elif number < 0:
17         negative()
18     else:
19         print('Я вас не совсем понял. ;)')
20
21
22
23
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  JUPYTER: VARIABLES
PS F:\универ\программирование\8\berd8> & "F:/visual_code/python.exe" "f:/универ/программирование/8/berd8/Задание 1.py"
Введите число: 21
Положительное
PS F:\универ\программирование\8\berd8> |
```

Рисунок 1. Работа программы
«Задание 1»



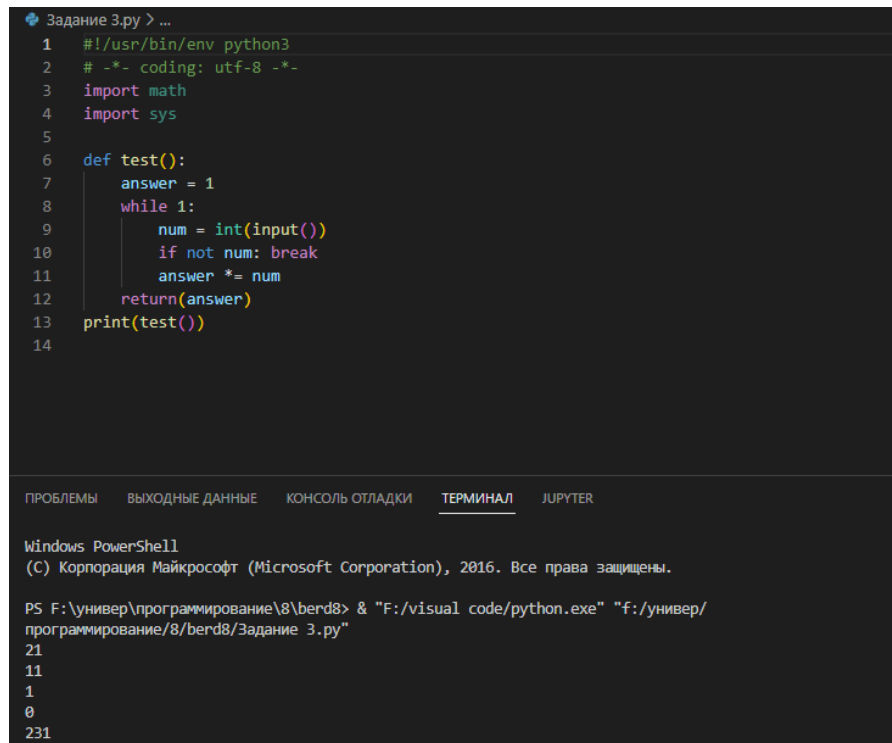
```
Задание 1.py > ...
7  number = int(input('Введите число: '))
8  if number > 0:
9      positive()
10 elif number < 0:
11     negative()
12 else:
13     print('Я вас не совсем понял. ;)')
14     test()
15
16 def negative():
17     print('Отрицательное')
18
19 def positive():
20     print('Положительное')
21
22 test()
23
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  JUPYTER: VARIABLES
PS F:\универ\программирование\8\berd8> & "F:/visual_code/python.exe" "f:/универ/программирование/8/berd8/Задание 1.py"
Введите число: 21
Положительное
PS F:\универ\программирование\8\berd8> |
```

Рисунок 2. Работа программы
«Задание 1»

Нет не имеет порядка определения функций. Функция при

необходимости может получать и возвращать данные.

Задание 2.



```
Задание 3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  import sys
5
6  def test():
7      answer = 1
8      while 1:
9          num = int(input())
10         if not num: break
11         answer *= num
12     return(answer)
13 print(test())
14
```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

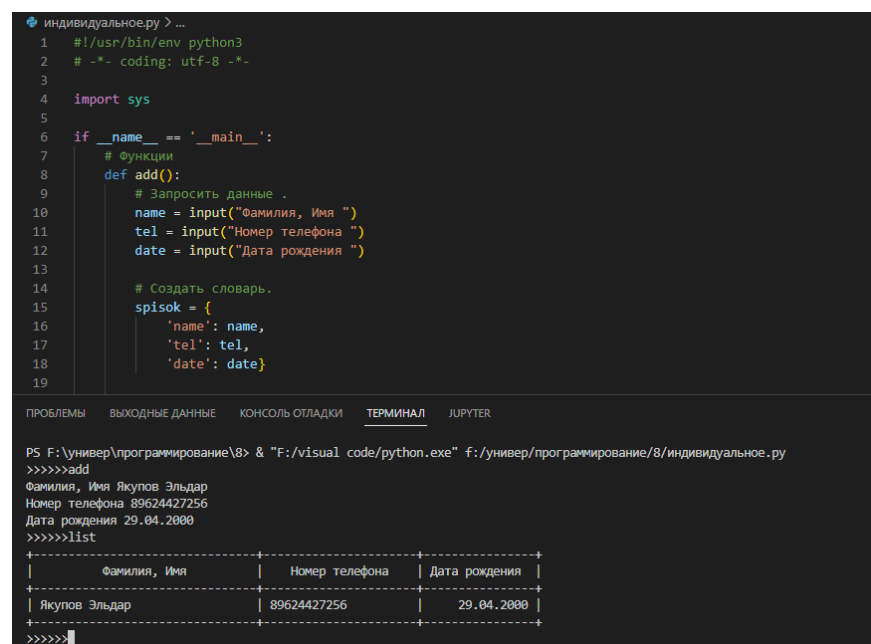
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation), 2016. Все права защищены.

PS F:\универ\программирование\8\berd8> & "F:/visual code/python.exe" "f:/универ/программирование/8/berd8/Задание 3.py"

21
11
1
0
231

Рисунок 3. Работа программы
«Задание 2»

Индивидуальное:



```
индивидуальное.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Функции
8      def add():
9          # Запросить данные .
10         name = input("Фамилия, Имя ")
11         tel = input("Номер телефона ")
12         date = input("Дата рождения ")
13
14         # Создать словарь.
15         spisok = {
16             'name': name,
17             'tel': tel,
18             'date': date
19         }
20
```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

PS F:\универ\программирование\8> & "F:/visual code/python.exe" f:/универ/программирование/8/индивидуальное.py

>>>>>add
Фамилия, Имя Якупов Эльдар
Номер телефона 89624427256
Дата рождения 29.04.2000
>>>>>list

Фамилия, Имя	Номер телефона	Дата рождения
Якупов Эльдар	89624427256	29.04.2000

>>>>>

Рисунок 4. Работа программы
«Индивидуальное»

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.

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` ?

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?

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?

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

5. Какие существуют способы передачи значений в функцию?

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?

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?

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?

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 строкдокументации?

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