Методические указания

Урок 8.1. Функции

Задачи урока:

- Научиться писать собственные функции

0. Подготовка к уроку

До начала урока преподавателю необходимо:

1) Просмотреть, как ученики справились с домашним заданием
2) Прочитать методичку

1. Функции без параметров

Учитель: Сегодня поработаем все вместе. В предыдущих уроках мы использовали встроенные в Python функции `print()`, `input()`, `int()`, `str()`, `len()` и многие другие. Пришло время начать писать свои собственные функции.

Предположим нам надо написать программу, которая рисовала бы звездный прямоугольник размерами 5×7 (5 строк и 7 столбцов).

Наш первый вариант кода выглядел бы примерно так:

```
print('*******')
print('*******')
print('*******')
print('*******')
print('*******')
```

Далее мы изучили оператор умножения строки на число (оператор повторения) и написали бы код:

```
print('*' * 7)
print('*' * 7)
print('*' * 7)
print('*' * 7)
print('*' * 7)
```

Ну и, наконец, мы изучили циклы, после чего код принял бы вид:

```
for _ in range(5):
    print('*' * 7)
```

А теперь представим, что таких прямоугольников нужно изобразить несколько, скажем, три штуки.
Тогда код программы будет иметь вид:

```
for _ in range(5):
    print('*' * 7)
print()
for _ in range(5):
    print('*' * 7)
print()
for _ in range(5):
    print('*' * 7)
```

Результатом выполнения такого кода будет:

```
*******
*******
*******
*******
*******

*******
*******
*******
*******
*******

*******
*******
*******
*******
```

```
*******
```

Этот код полностью решает поставленную задачу, но не лишен недостатков.

- Во-первых, он довольно громоздок, поскольку часть кода отвечающая за вывод прямоугольника дублируется.
- Во-вторых, если понадобится поменять размеры прямоугольника, придется менять их столько раз, сколько раз он выводится.

Во избежание таких недостатков используют функции. Вместо повторения кода, осуществляющего вывод прямоугольника, перенесем его в отдельную функцию, а затем вызовем ее 3 раза.

Для создания функции пишем такой код:

```
def draw_box():
    for _ in range(5):
        print('*' * 7)
```

Когда функция создана, чтобы увидеть результат ее работы, нужно ее вызвать по имени, следующим образом:

```
draw_box()
```

Теперь для того, чтобы изобразить 3 прямоугольника мы напишем код:

```
draw_box()
print()
draw_box()
print()
draw_box()
```

Можно вызывать функцию в цикле столько раз, сколько нужно.

Код стал короче, читабельнее (за счет удачного названия функции), а главное, если нам потребуется изменить размеры прямоугольника, то достаточно будет внести изменения только в самой функции `draw_box()`.

Имена функциям назначают точно так же, как переменным. Имя функции должно быть достаточно описательным, чтобы любой, читающий ваш код, мог легко догадаться, что именно функция делает. Python требует соблюдения тех же правил, что при именовании переменных. Поскольку функции выполняют действия, большинство программистов предпочитает в их именах использовать глаголы.

Итак, функция – отдельная, независимо действующая часть программы, выполняющая определенную задачу. Функции объявляются с помощью ключевого слова def (от define – определять). За ключевым словом def следует название функции, круглые скобки () и двоеточие.

```
def название_функции():
    блок кода
```

Первая строка объявления функции называется ее заголовком. Начиная со следующей строки идет блок кода - тело функции, набор инструкций, составляющих единое целое. Эти инструкции выполняются каждый раз, когда вызывается функция.

Обратите внимание, каждая строка в теле функции выделена отступом. Последняя выделенная отступом строка - последняя строка тела функции. Для выделения строк блока кода отступом программисты Python обычно используют четыре пробела, в соответствии со стандартом PEP 8.

Рассмотрим объявление функции:

```
def print_message() :
    print('Я - Артур,')
    print('король британцев. ')
```

Этот фрагмент кода определяет функцию с именем `print_message()`. Тело функции состоит из двух инструкций. Вызов функции приведет к исполнению этих инструкций.

После того, как функция объявлена, чтобы увидеть ее результат, нужно ее вызвать - написать ее название, а затем круглые скобки.

```
# объявление функции
```

```
def print_message() :
    print('Я - Артур,')
    print('король британцев. ')

# вызов функции
print_message()
```

2. Функции с параметрами

Учитель: До этого мы определили функцию draw_box(), которая выводит звездный прямоугольник с размерами 5×7:

```
def draw_box():
    for i in range(5):
        print('*' * 7)
```

Было бы намного удобнее, если бы функция draw_box() выводила прямоугольник с произвольными размерами. И действительно, функции могут принимать параметры, что делает их более гибкими.

Функции с параметрами объявляют также, как и функции без параметров, только в скобках указывают параметры:

```
def название_функции(параметры):
    блок кода
```

Давайте перепишем предыдущую версию функции draw_box() так, чтобы она принимала параметры, задающие высоту и ширину прямоугольника:

```
def draw_box(height, width):    # функция принимает два параметра
    for i in range(height):
        print('*' * width)
```

Теперь наша функция draw_box() принимает два целочисленных параметра height – высота прямоугольника и width – ширина прямоугольника, и для ее вызова нам нужно обязательно их указать.

Чтобы вывести звездный прямоугольник размерами 5 на 7 мы пишем код:

```
draw_box(5, 7)
```

Результатом такого вызова функции draw_box(5, 7) будет:

```
*******
*******
*******
*******
*******
```

Чтобы вывести прямоугольник размерами 10 на 15, мы пишем код:

```
draw_box(10, 15)
```

Результатом такого вызова функции draw_box(10, 15) будет:

```
***************
***************
***************
***************
***************
***************
***************
***************
***************
***************
```

Теперь с помощью нашей новой версии функции draw_box() можем в одной программе выводить прямоугольники разных размеров. Следующий программный код:

```
draw_box(3, 3)
print()
draw_box(5, 5)
print()
draw_box(4, 10)
```

выведет:

```
***
***
***

*****
*****
*****
*****
*****

**********
**********
**********
**********
```

На место параметров мы можем подставлять не только целочисленные константы, но и значения переменных. Следующий программный код:

```
n = 3
m = 9
draw_box(n, m)
```

выведет:

```
*********
*********
*********
```

4. Решение задач

Задача 1

Написать функцию, которая принимает стороны прямоугольника и выводит периметр и площадь прямоугольника

Решение

```
def func(side1, side2):
    area = side1 * side2
    perimetr = (side1 + side2) * 2
    print(area, perimetr)
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1