Ссылка на урок:

https://coreapp.ai/app/player/lesson/6714b04fcf2b73693a23e96b (для учеников).

Занятие 5

Тема. Вывод данных в Python

1. Теоретическая часть

Мы уже познакомились с типами данных и рассмотрели оператор присваивания. Этого достаточно для того, чтобы записать программу преобразования данных. Но результат этих преобразований нам виден не будет. Так как для вывода данных из оперативной памяти на экран компьютера нужно использовать *оператор вывода* (встроенную функцию) **print().**

Общий вид оператора:

print (<выражение 1>, <выражение 2>, <выражение N>).

Здесь в круглых скобках помещается *список вывода* (список передаваемых аргументов) – список выражений, значения которых выводятся на экран. *Под выражениями (аргументами) здесь понимаются числовые, символьные и логические выражения, в том числе константы и переменные.*

К сведению! В Python 2 скобки можно было не указывать, но в Python 3 скобки необходимы. Всё потому, что в третьей версии `print()` — это ещё и функция, а не только инструкция.

Примеры:

```
>>> a = 5
>>> b = 10
>>> c = 2
>>> print((a+b)*c)
30
>>>
```

```
>>> x=3
>>> y=6
>>> z=-10
>>> print(x,y,z)
3 6 -10
>>>
```

Обратите внимание: по умолчанию выводимые выражения разделяются одним пробелом. Иначе говоря, пробел является *разделителем* (сепаратором, от англ, *separator*) между выводимыми выражениями. Но разделителем может быть и другой символ. Тогда этот символ в списке аргументов функции *print* нужно указать с помощью аргумента **sep**. Разделитель можно совсем убрать. Можно каждое выражение из списка выводить с новой строки. Всё это также делается с помощью аргумента **sep,** у которого есть и другие возможности.

Вариант организации вывода	Оператор (функция) вывода	Результат
По умолчанию	print (1, 20, 300)	1 20 300
Убрать разделители-пробелы (поставить две одинарные кавычки рядом)	print(l, 20, 300, sep='')	120300
Добавить разделитель-запятую	print(1, 20, 300, sep=',')	1, 20, 300
Вывод каждого значения с новой строки	print(l, 20, 300, sep='\n')	1 20 30

Вывод может быть с ***комментариями (поясняющим текстом),*** который заключается в кавычки, например, двойные:

```
>>> x=3
>>> y=6
>>> z=-10
>>> print("x=",x,"y=",y,"z=",z)
x= 3 y= 6 z= -10
>>>
```

или одинарные:

```
>>> a = 100000
>>> b = 100000
>>> print ('a * b =', a*b)
a * b = 10000000000
>>>
```

По умолчанию каждая команда **print** осуществляет вывод с новой строки, *например:*

```
>>> x = 1.6 * 2
>>> y = x * 2
>>> z = x + y
>>> print (x)
3.2
>>> print (y)
6.4
>>> print (z)
9.600000000000001
>>>
```

Чтобы убрать переход к новой строке, используется аргумент **end**=“” (или end=’’), например:

```
>>> print('2', end=''); print('6', end='')
26
```

Форматный (форматированный) вывод данных

В данных примерах мы использовали целые числа (**int**). Здесь всё просто. Python, в отличии от многих других ЯП, может работать с произвольно большими целыми числами, если для их поддержки достаточно памяти. Но чаще приходится работать с вещественными числами (**float).**

При выводе вещественных чисел по умолчанию выводится **16 знаков после запятой**. Не всегда такой формат необходим. Если нужно сократить формат вывода, то используется *форматный вывод* или *F-строки*. Форматный вывод применяется также в некоторых случаях и для целых чисел, и для строкового типа (**string**).

В форматном выводе данных используется встроенная функция **format()**. Общий вид оператора**:**

print ("{}, {}, … ,{}".format(a,b))

Здесь:
- справа от точки – функция format (a,b), аргументами которой являются те величины (a,b), которые выводятся;
- символьная строка в кавычках, слева от точки, – это форматная строка, которая определяет **формат вывода** (то есть, как именно величины будут представлены на экране);
- фигурные скобки внутри форматной строки содержат список вывода в нужной форме.

Для вывода вещественного числа в списке вывода для каждого выражения указываются два параметра:
1) общее количество позиций, отводимых на число;
2) количество позиций в дробной части числа.

Примеры.

```
>>> a = 1 / 3
>>> print ("{:7.3f}".format(a))
  0.333
>>> b = 1 / 9
>>> print ("{:7.4f}".format(b))
 0.1111
```

Если цифр в числе меньше, чем зарезервированных под него позиций на экране, то свободные позиции дополняются пробелами слева от числа.

Можно объединить оба вывода в одном:

```
>>> a = 1 / 3; b = 1 / 9
>>> print ("{:7.3f}{:7.4f} ".format(a, b))
  0.333 0.1111
>>>
```

А вот это пример того, как можно с помощью оператора (функции) print вывести на экран выражение с ответом. Здесь форматный вывод удобен тем, что его можно использовать в общем виде, не привязываясь к конкретным значениям a, b.

```
>>> a=12
>>> b=13
>>> c=a+b
>>> print("{}+{}={}".format(a, b, c))
12+13=25
>>>
```

2. **Просмотр учебного видеоролика «Обучение Python. Вывод данных. Команда Print»** (с целью расширения знаний, наглядного представления и закрепления материала):

https://rutube.ru/video/11a825bba2bae39f8d98684fa36a714b/?r=plwd (длина ролика: 10:01).

3. **Закрепление знаний**

ЗАДАНИЕ 1.

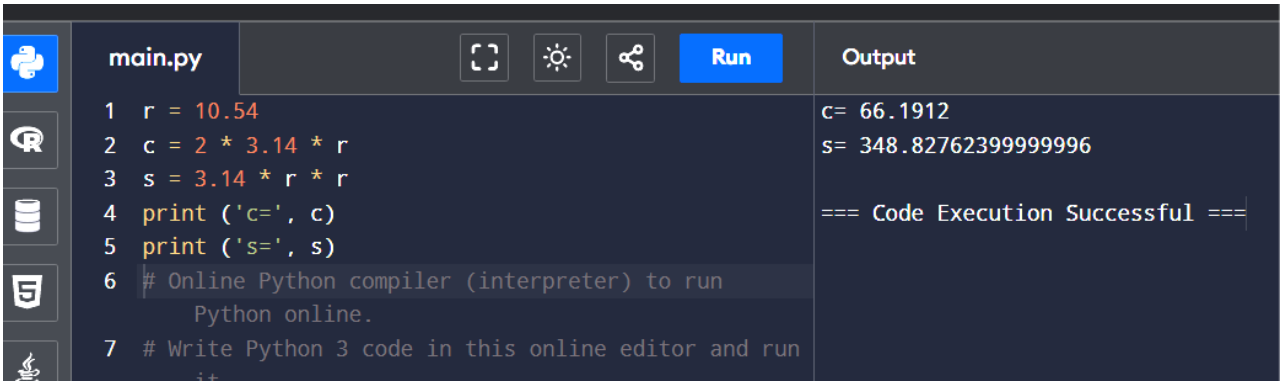
Напишите и выполните программу на платформе Programiz.com, вычисляющую длину окружности и площадь круга, если известен радиус. Значение числа PI принять равным 3.14, значение радиуса задать равным 10.54 (см).

Исходные данные и результаты связаны соотношениями, известными из курса математики: $c = 2\pi r$, $s = \pi r^2$.

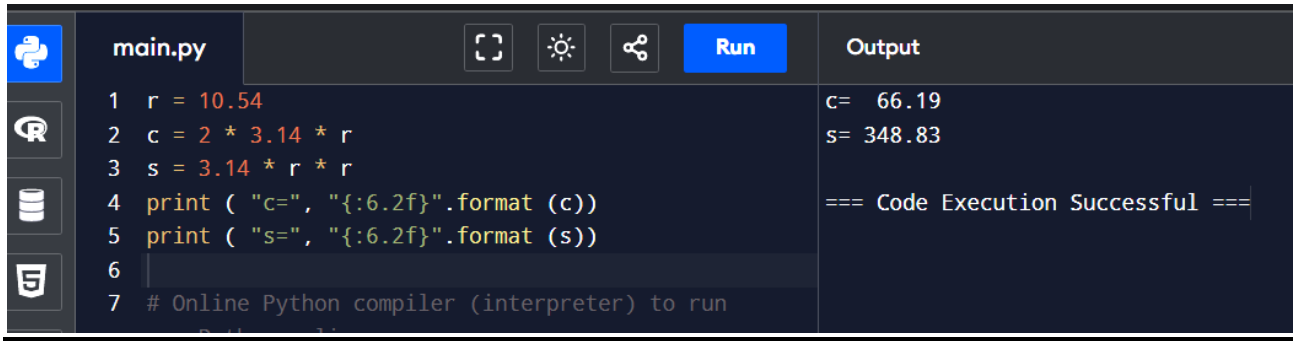
Дополнительное задание. Внесите изменения в оператор *print*, чтобы результат выводился с двумя знаками после запятой.

! Решение будет на следующей странице

Решение (основное задание):



Решение (дополнительное задание):



ЗАДАНИЕ 2. (Вопрос с автопроверкой)

Для каждого оператора **print ()** запишите соответствующий ему результат работы:

а) print (10, 20, 30)

б) print(10, 20, 30, sep='')

в) print(10, 20, 30, sep=', ')

г) print (10, 20, 30, sep=':')

д) print(10, 20, 30, sep=',')

Ответ.

а)	print (10, 20, 30)	4) 10 20 30
б)	print(10, 20, 30, sep='')	1) 102030
в)	print(10, 20, 30, sep=', ')	2) 10, 20, 30
г)	print (10, 20, 30, sep=':')	3) 10:20:30
д)	print(10, 20, 30, sep=',')	5) 10,20,30