The Print Function

All Versions

Examples

Print basics

In Python 3 and higher, `print` is a function rather than a keyword.

```
print('hello world!')
# out: hello world!

foo = 1
bar = 'bar'
baz = 3.14

print(foo)
# out: 1
print(bar)
# out: bar
print(baz)
# out: 3.14
```

You can also pass a number of parameters to `print` :

```
print(foo, bar, baz)
# out: 1 bar 3.14
```

Another way to `print` multiple parameters is by using a `+`

```
print(str(foo) + " " + bar + " " + str(baz))
# out: 1 bar 3.14
```

What you should be careful about when using `+` to print multiple parameters, though, is that the type of the parameters should be the same. Trying to print the above example without the cast to `string` first would result in an error, because it would try to add the number `1` to the string `"bar"` and add that to the number `3.14` .

```
# Wrong:
# type:int  str  float
print(foo + bar + baz)
# will result in an error
```

This is because the content of `print` will be evaluated first:

```
print(4 + 5)
# out: 9
print("4" + "5")
# out: 45
print([4] + [5])
# out: [4, 5]
```

Otherwise, using a `+` can be very helpful for a user to read output of variables In the example below the output is very easy to read!

The script below demonstrates this

```
import random
#telling python to include a function to create random numbers
randnum = random.randint(0, 12)
#make a random number between 0 and 12 and assign it to a variable
print("The randomly generated number was - " + str(randnum))
```

You can prevent the `print` function from automatically printing a newline by using the `end` parameter:

```
print("this has no newline at the end of it... ", end="")
print("see?")
# out: this has no newline at the end of it... see?
```

If you want to write to a file, you can pass it as the parameter `file` :

```
with open('my_file.txt', 'w+') as my_file:
    print("this goes to the file!", file=my_file)
```

> this goes to the file!

Syntax

Parameters

Remarks