Python

The print() function in python

How print() Function works in Python?

The Python print() function by default displays to the standard console. print() without any arguments displays the new line to the console hence when you print a string, it displays a string and also adds a new line to the console.

1. Syntax of print()

Following is the syntax of the print() statement.

Syntax of print( )

```
print(*objects, sep='  ', end='\n', file=sys.stdout, flush=False)
```

Python Print Function Parameters

- ☐ Objects
- ☐ sep
- ☐ end
- ☐ file
- ☐ flush

1.1 Parameters of print()

- `objects` – The object you wanted to print. It displays the content of the object to standard output. You can use this to print multiple objects by using a comma separator.
- `sep` – Used to specify how to separate the object.
- `end` – Used the character to used at the end of the printed object. Default sets to \n (newline)
- `file` – Specify the file object where you wanted to print the object. Default is standard console which is sys.stdout.
- `flush` – Specify whether to buffer the output or flush to console.

2. Python print() Function Example

First, let's see the default behavior of the Python print() function that just takes the text.

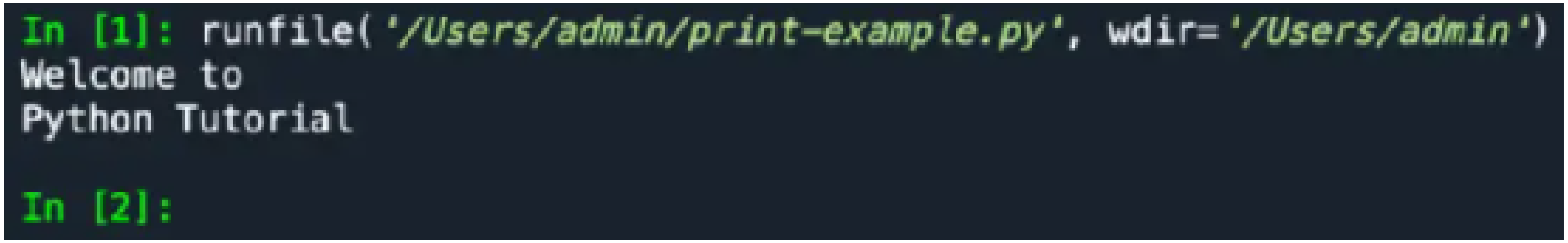
By default when you display text in Python by using the print() function, each text issued with a print function writes in a new line. Here, is an example.

```
# print() usage

print("Welcome to")

print("Python Tutorial")
```

This code yields the following output on the console.



```
In [1]: runfile('/Users/admin/print-example.py', wdir='/Users/admin')
Welcome to
Python Tutorial

In [2]:
```

3. Print Multiple Texts

The print() function can also be used to print multiple statements, all you need to do is pass all the objects you wanted to print as an argument to the function.

```
# Print multiple strings

str1 = "Welcome to"

str2 = "Python Tutorial"

print(str1,str2)
```

This code yields the following output on the console.

```
# Output:

# Welcome to Python Tutorial
```

4. Print using sep Parameter

The sep parameter is used to separate the objects on the console by the specified separator

```
# Print multiple strings

str1 = "Welcome to"

str2 = "Python Tutorial"

print(str1,str2, sep=",")
```

This code yields the following output on the console.

```
 # Output:

 # Welcome to,Python Tutorial
```

5. Print using end Param

As you saw above the print() function by default prints objects in a new line. To print without a newline you can use the `end` param. By using this you can specify what delimiter you wanted to use when printing on a single line.

```
# Printing text without newline
print("Welcome to", end =" ")
print("Python Tutorial")
```

This code yields the following output on the console.

```
# Output:
Welcome to Python Tutorial
```

6. print() Function with file Parameter

By default print() function prints the object to the Python standard console which is sys.stdout. You can change this to print it to the file by using the `file` parameter.

```
# Using file param
logSourceFile = open('logfile.txt', 'w')
print("Welcome to Python Tutorial", file = logSourceFile)
logSourceFile.close()
```

Here, the `open()` method is used to open the `logfile.txt` file and `w` param specified to open file in write mode. If a file doesn't exists it create a new file and opens in write mode.

We also used the `file` param with the value `logSourceFile` to the print() function. hence, it writes the text to the logfile.txt.

After completing on writing, may sure you close the file object by using `close()` method.

7. Print() with flush Parameter

By default the value to the `flush` param is set to `False` meaning it waits for the line to complete before printing on console.

If you set fluash param to True, it flushes the text to console as it receives. When using this on small text you may not be much difference.

Note that the lush param doesn’t affect the output instead it just defines how fast the date should be printed on the console.

```
# Using flush param

fruits={'apple','mango','guava',}

for x in fruits:

    print(x, end=" ", flush=True)
```