print()

Python Print Function

Function	Description
`print([data])`	Prints the data argument to the console followed by a new line character. If the call doesn't include a data argument, this function prints a blank line to the console.

print() function:

A function is a reusable unit of code that performs a specific task like getting input from users or printing output to the console. Python provides several built-in functions including the print() function. The print function displays the data that is passed to it as arguments. The function starts with the function name and is followed by parenthese. Within the parenthese is the data attributes. In the example below, "Hello world!" is the attribute for the function. See the results after the code section below.

In [1]:
```
print("Hello world!")
```

```
Hello world!
```

Within the double-quotes, almost anything can be included. As long as it is alphanumeric.

In [2]:
```
print("#$%^&*_~~sdaa12+34567890_AbCdEfGhI,j,K,l,M,nOpQrStUvWxYz")
```

```
#$%^&*_~~sdaa12+34567890_AbCdEfGhI,j,K,l,M,nOpQrStUvWxYz
```

> **Tip:** Excape characters are used to add special characters, commas ',' are used to join lines together, and concatenators '+' are used to merge two items together. These advanced topics is discussed later.

Printing Numbers

Remember, anything within the " " will be printed.

In [3]:
```
print("3+7")
```

```
3+7
```

To use math operations within a print() function remove the double-quotes.

In [4]:
```
print(3+7)
```

```
10
```

Printing variables

Variables within a print statement can be sepparated different ways. A concatenator '+' or a comma ','

Notice the difference in the output in each example.

In [5]:
```
myName = "Dave"        # String
myAge = "32"           # String

print(myName + myAge)
```

```
Dave32
```

In [6]:
```
myName = "Dave"        # String
myAge = "32"           # String

print(myName, myAge)
```

```
Dave 32
```

In [7]:
```
myName = "Dave"        # String
myAge = 32             # Integer

print(myName, myAge)
```

```
Dave 32
```

> **Warning:** Be aware of using the concatenator and variables of different types. It is an easy problem to solve by casting. (i.e., str(myAge))

In [8]:
```
myName = "Dave"        # String
myAge = 32             # Integer

print(myName + myAge)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-80ea5a0ec167> in <module>
      2 myAge = 32             # Integer
      3
----> 4 print(myName + myAge)

TypeError: can only concatenate str (not "int") to str
```

Casting

If the myAge variable is casted as a string then the code from the previous example will work.

In [9]:
```
myName = "Dave"        # String
myAge = 32             # Integer

print(myName + str(myAge))
```

```
Dave32
```

> **Tip:** Casting gives you the ability to change the data type to the type you want. Three types of casting functions:

int() - constructs an integer number from an integer, a float, or a string

float() - constructs a float number from an integer, a float or a string (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer and float

---

Resources used and great for deeper research:

1. W3Schools.com
2. BDM Publications
3. Real Python - Guide to the print() function