In [64]:
```
# 1.1 Running Cells and Displaying Output
# This cell is provided in Part 1: Using JupyterLab
"Will this line be displayed?"
print("Hello" + ",", "world!")
5 + 3
```

```
Hello, world!
```

Out[64]:
```
8
```

In [65]:
```
# 1.2 Viewing Documentation
# help() operation used to display command documentation
help(print)
```

```
Help on built-in function print in module builtins:

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

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

In [66]:
```
# Shift + Tab hotkey in order to show documentation without help() command
print('Welcome.')
```

```
Welcome.
```

In [67]:
```
# 1.3 Importing Libraries
# Importing common Python libraries that will be used in this assignment
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# 2.1 Summation Function
# Below is a function summation that evaluates the values from i=1 to n for (i^3 + 3i^2
n = input("What is the value of n? (>= 1): ")
n = int(n)

if n >= 1:
    def summation(start, end, expression):
        return sum(expression(i) for i in range(start, end))

    def function(i):
        return i ** 3 + 3 * (i ** 2)

    print("i^3 + 3i^2 =")
    print(summation(1, n+1, function))

else:
    print("n is too small!")
```

In [ ]:
```
# 2.2 list_sum Function
# Takes input list_1 and list_2 (same # of elements)
# Squares of list_1 values, Cubes of list_2 values, and a third list with the sum of bo

def list_sum(list_1: list, list_2: list) -> list:
    list_3 = [] # to be allocated with array addition in for loop
    list_length = len(list_1)
    for i in range (list_length):
        arrSums = (list_1[i]**2) + (list_2[i] **3)
        list_3.append(arrSums)
    return list_3
```

In [ ]:
```
# 3.1 NumPy Array Creation
# Initalize two NumPy Arrays (arr1 = 1,2,3) (arr2= 4,5,6)
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# 3.2 NumPy Array Addition
# Adds two different arrays using NumPy's add function
np.add(arr1,arr2)
```

In [ ]:
```
# 3.3 NumPy array_sum Function Recreation
# Takes input list_1 and list_2 (same # of elements)
# Squares of list_1 values, Cubes of list_2 values, and a third list with the sum of bo
# Now using numpy functions

import numpy as np

def array_sum(list_1: np.array, list_2: np.array) -> np.array:
    squares = np.square(list_1)
    print(squares)

    cubes = np.power(list_2, 3)
    print(cubes)

    return np.add(squares,cubes)
```

In [ ]:
```
# 3.4 Runtime Comparision using %%time
# Comparision between the list_sum (Python 3) and array_sum (NumPy) methods
# By running randomly generated lists/arrays on these methods, and using %%time to see
# it can be determined that array_sum is faster, running in the microseconds vs list_su
```

In [ ]:
```
%%time

sample_list = list(range(100000))

def list_sum(list_1: list, list_2: list) -> list:
    list_3 = [] # to be allocated with array addition in for loop
    list_length = len(list_1)
    for i in range (list_length):
        arrSums = (list_1[i]**2) + (list_2[i] **3)
        list_3.append(arrSums)
```

```
    return list_3

list_sum(sample_list, sample_list)
```

In [ ]:

```
%%time

import numpy as np

sample_array = np.arange(100000)

def array_sum(list_1: np.array, list_2: np.array) -> np.array:
    squares = np.square(list_1)
    print(squares)

    cubes = np.power(list_2, 3)
    print(cubes)

    return np.add(squares,cubes)

array_sum(sample_array, sample_array)
```

In [ ]:

```
# 4.1 Function Graphing using MatPlotLib
# Graphing f(t) = 3 sin(2*pi*t). x limit is [0,pi]. y limit is [=10,10].

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,2 * np.pi, 0.1)
y = np.sin(2 * np.pi * x)

xValues = [0, np.pi/2, np.pi]
labels = ['0', 'π/2', 'π']

plt.axes(xlim=(0, 3), ylim=(-10, 10))
plt.xticks(xValues, labels)

plt.plot(x,3*y, 'r+')

plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.title('Sine Function: 3sin(2πt)')

plt.show()
```