Python Reference Sheet

Contents

Chapter 1

Introduction

Python is a high-level, general purpose programming language which can be used in a wide range of fields. It is very user-friendly as the syntax can be easily understood. *Note: Python is a good programming language to learn as a stepping stone, however if you would like to pursue a long-term path in programming, you should take a look at C++ and Java since the syntax there is more complex than in Python.* This reference sheet is for students interested in the basics of Python programming so that you can apply them to the programming exercises on the site. If you are interested in Python learning resources past this reference sheet, take a look at the list of links below:

- Getting Started With Python: python.org
- python.org Documentation: python.org
- Python w3schools Tutorials: w3schools.com
- w3schools Reference: w3schools.org
- freeCodeCamp.org Python for Beginners freeCodeCamp.org
- Microsoft Python for Beginners microsoft.com
- geeksforgeeks.org Python Tutorial geeksforgeeks.org

Chapter 2

Printing in Python

To print anything on your screen in Python, whether that be a message, question, or number all you have to use is the `print()` function. In a terminal on your computer, you would simply write something like this:

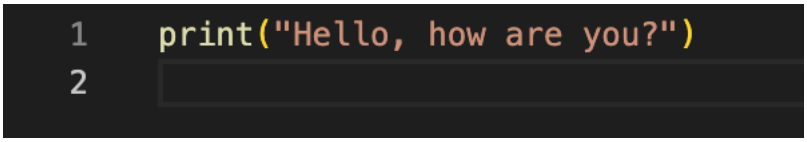

Figure 2.1: Using a print function in Python

Your output would look like the following:

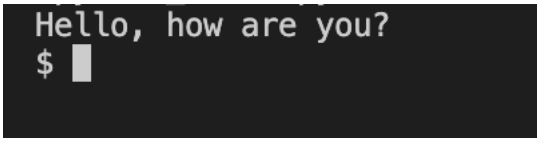

Figure 2.2: Print function output in Python

Click the link below to do a few practice exercises:

Printing in Python Exercises: Printing a number in Python

If you would like to learn more about printing in Python, consider taking a look at the following links:

- Python Print Function w3schools.com
- Output using print() function geeksforgeeks.org
- Input and Output python.org

Chapter 3

Assigning values to variables in Python

In Python, instead of directly passing values into `print()` statements, we can create variables that store variables which can make your code look cleaner. In order to assign a value to a variable, you first need to create a variable. To create a variable in Python, you simply choose a character or group of characters that has some meaning to the value you are assigning it to. For example, you can declare a variable `x`, `symbol`, and `message`. Those variables can hold corresponding values in them. `x` holds the value `5.8`, `symbol` holds the value `'%'`, and `message` holds the value `"Hello World!"`. As you can see these variables have a valid meaning to the values assigned to them. In a terminal on your computer, you would simply declare a variable like so:

```
1   x = 5.8
2   symbol = '%'
3   message = "Hello World!"
4
5   print(x, symbol, message)
6
```

Figure 3.1: Using variable declaration in Python

Your output would look like the following:

```
5.8 % Hello World!
$
```

Figure 3.2: Variable declaration output in Python

Click the link below to do a few practice exercises:

Variable declaration in Python Exercises: Assigning variables in Python

If you would like to learn more about assigning variables in Python, consider taking a look at the following links:

- Python Variables w3schools.com
- Python Variables geeksforgeeks.org
- Variables and Types learnpython.org

Chapter 4

Writing Functions in Python

If you have a complicated program involving many calculations, and you want to write code in an organized fashion, you can create functions that performs small tasks and that can be called by a main program function to run the entire program smoothly. For example if you would like to create a program that performs arithmetic operations between two numbers a and b, those being addition (+), subtraction (-), division (÷), and multiplication (×). You can create a function for each of those operations in your program to structure your code. Functions are useful because if you would like to expand your arithmetic program to something like a mathematical library with many operations, each operation would have its own function. In a terminal on your computer, you can create a function like so:

```
1  def addition(x, y):
2      return x + y
3
4  if __name__ == '__main__':
5      x = 5
6      y = 6
7
8      print("The output is: " + str(addition(x, y)))
9
```

Figure 4.1: Using functions in Python

Your output would look like the following:

```
The output is: 11
$
```

Figure 4.2: Function output in Python

Click the link below to do a few practice exercises:

Functions in Python Exercises: Functions in Python

If you would like to learn more about functions in Python, consider taking a look at the following links:

- Python Functions w3schools.com
- Python Functions geeksforgeeks.org

Chapter 5

User Inputs in Python

Instead of hard-coding values to variables in Python programs, what you can do is have the user input a value which is then assigned to a variable within the program. The program can then continue to execute based on the user's interaction with the program. To ask for user input in Python, you first declare the type of input the user will enter into the program, followed by using the `input()` function. In a terminal on your computer, you can create single user input like so:

```
1   number = float(input("Enter a number: "))
2   character = input("Enter a character: ")[0]
3   message = str(input("Enter a string: "))
4
5   print("Number:    ", str(number))
6   print("Character: ", str(character))
7   print("Message:   ", str(message))
8
```

Figure 5.1: Single user input in Python

Your output would look like the following:

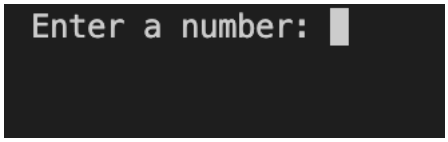

Figure 5.2: Program waiting for user input in Python

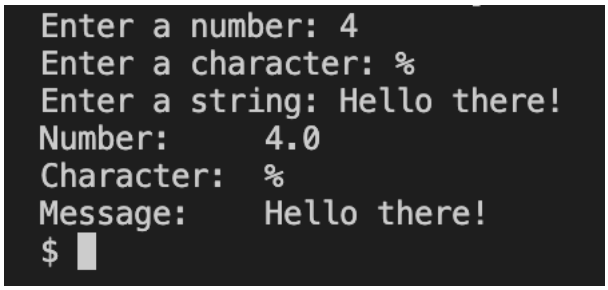

Figure 5.3: Program output after all user inputs

Click the link below to do a few practice exercises:

Single input in Python Exercises: Single Input in Python

If you would like to learn more about inputting singular values in Python, consider taking a look at the following links:

- Python input() w3schools.com
- Python input() geeksforgeeks.org

Python also allows the user to input more than one input in a single line, storing all of those inputs separately. This is is useful, for example, if someone would like to create a program that would store and list out names in alphabetical order. For multiple inputs, we need to write some code in order to store this information. This can be done through the use of arrays `[]`. In order to print out all the names as an output, we also have to use a `for loop`. A `for loop` allows programmers in Python to iterate through code, and in the case of arrays, iterate through the array and output each element value contained within the array. In a terminal on your computer, you can create multiple user input like so:

```
1  names_input = input("Enter names separated by commas: ")
2  names = [name.strip() for name in names_input.split(",")]
3
4  sorted_names = sorted(names, key=str.lower)
5
6  print("Alphabetical names:")
7  for name in sorted_names:
8      print(name)
9
```

Figure 5.4: Multiple user input in Python

Your output would look like the following:

```
Enter names separated by commas: George, Tom, Alice, Zack, Peter
Alphabetical names:
Alice
George
Peter
Tom
Zack
$
```

Figure 5.5: Program output after user input

Click the link below to do a few practice exercises:

Multiple input in Python Exercises: Multiple Input in Python

If you would like to learn more about inputting multiple values in Python, consider taking a look at the following links:

- How to input multiple values from user in one line in Python? tutorialspoint.com
- Taking multiple inputs from user in Python geeksforgeeks.org

Chapter 6

Plotting in Python

Python differs from other programming languages in that it has a built-in graphing/plotting library allowing users to output two or three dimensional graphs which is great for visualization, especially for something like recording the trends of physical phenomena or simply visualizing a three dimensional shape. In order to plot in two and three dimensions in Python, we use the library `matplotlib.pyplot`. When creating a plotting program in python the basic formatting of code includes the following:

```
1   import matplotlib.pyplot as plt
2
3   fig, ax = plt.subplots()
4
5   ax.set_xlim(-10, 10)
6   ax.set_ylim(-10, 10)
7   ax.set_xlabel('x-axis')
8   ax.set_ylabel('y-axis')
9   ax.set_title('Basic Plot in Python')
10  ax.grid(True)
11
12  plt.savefig('BasicPlot.svg')
13  plt.show()
14
```

Figure 6.1: 2D plot code in Python

Your output would look like the following (check figure 6.2):

The first line of code simply imports the plotting library, `matplolib.pyplot`. Then you have features which you can include in your plot, such as setting the x and y limits, setting the x and y axis titles, setting the title of the graph, setting the title for the plot, as well as saving the plot in any format you like. `plt.show()` is the command that allows you to see your plot output. Although not in the program above, to plot points, lines, or any shape in Python, you would use the `ax.plot()` function.

Click the link below to do a few practice exercises:

2D Plots in Python Exercises: 2D Plotting in Python

If you would like to learn more about plotting in 2D in Python, consider taking a look at the following

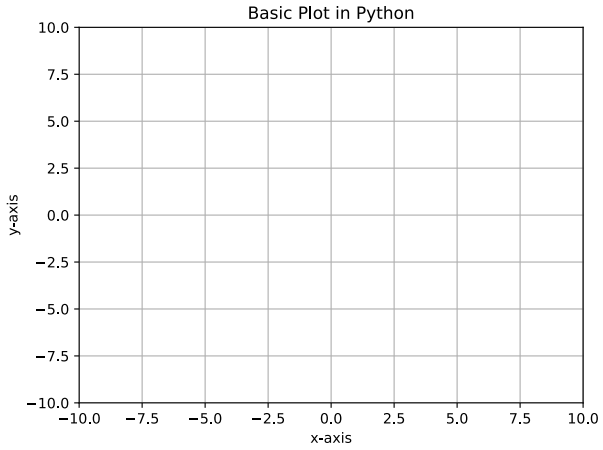

Figure 6.2: 2D Plot output in Python

links:

- Matplotlib Plotting w3schools.com
- Pyplot tutorial matplotlib.org

Plotting in 3D is slightly different as another package library needs to be introduced in the code. That package is `mpl_toolkits.mplot3d`. This package is what enables three dimensional plots. When creating a plotting program in python the basic formatting of code includes the following (check figure 6.3):

Your output would look like the following (check figure 6.4):

The first two lines of code simply imports the two plotting libraries, `matplolib.pyplot` and `mpl_toolkits.mplot3d`. Then you have most of the same features from the 2D plot code which you can include in your 3D plot code, such as setting the z limit, setting the z axis title, and so on. Although not in the program above, same as with plotting in 2D, to plot points, lines, or any shape in Python, you would use the `ax.plot()` function.

Click the link below to do a few practice exercises:

3D Plots in Python Exercises: 3D Plotting in Python

If you would like to learn more about plotting in 3D in Python, consider taking a look at the following links:

- Three-dimensional Plotting in Python using Matplotlib geeksforgeeks.org
- 3D Plotting matplotlib.org

```python
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-10, 10)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('Basic Plot in Python')
ax.grid(True)

plt.savefig('BasicPlot.svg')
plt.show()
```

Figure 6.3: 3D plot code in Python

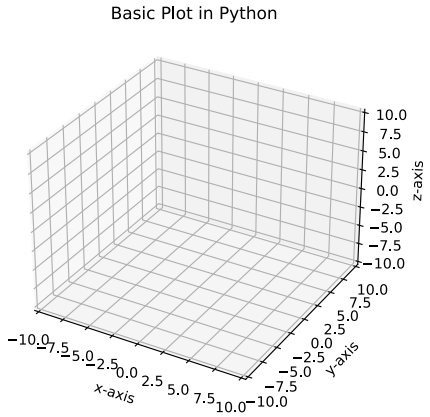

Figure 6.4: 3D Plot output in Python