

Python Basics

By @Thwarted_Pentester

List of Topics

- Variables
- Loops
- Functions
- Data Structures
- If Statements
- Files
- Import

#Topic-1 So Called basic "Hello World" Code

- Code

```
print("Hello World")
```

- Output

```
Python code output
```

```
Hello World
```

#Topic-2 **Mathematical Operators**

Like a calculator, there are operations such as adding, subtracting, multiplying, and dividing; using Python, we can code our calculator; after all, programming is just writing rules for the computer to follow given specific inputs and conditions. The table below shows the different operations:

Operator	Syntax	Example
Addition	+	1 + 1 = 2
Subtraction	-	5 - 1 = 4
Multiplication	*	10 * 10 = 100
Division	/	10 / 2 = 5
Modulus	%	10 % 2 = 0
Exponent	**	5**2 = 25 (5 ²)

Now that we know basic mathematical operators, let's move on to comparison operators; these play a big part in Python and will be built upon when we look at loops and if statements .

Symbol	Syntax
Greater than	>
Less than	<
Equal to	==
Not Equal to	!=
Greater than or equal to	>=
Less than or equal	<=

- Code of All Mathemetical Operators

```
print (21 + 43) # Addition
print (142 - 52) # Substraction
print (10 * 342) # Multiplication
print (10 / 2) # Division
```

```
print(5 ** 2) # Square
print ( 5 ** 3) # Cube
```

- Output

Python code output

```
64
90
3420
5
25
125
```

#Topic-3 **Variable & Datatypes

Variables

Variables allow you to store and update data in a computer program. You have a variable name and store data to that name.

- For e.g.: Create a variable "Hieght" and set it's initial value "200" now create a new variable and add "50" into it and now print the "Hieght"
- Code

```
height = 200
height = height + 50
print(height)
```

- Output

```
PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\variable.py"
250
PS D:\HACKING STUFFS\Programming Shit>
```

DataTypes

the type of data being stored in a variable. You can store text, or numbers, and many other types. The data types to know are:

- **String** - Used for combinations of characters, such as letters or symbols
- **Integer** - Whole numbers
- **Float** - Numbers that contain decimal points or for fractions
- **Boolean** - Used for data that is restricted to True or False options
- **List** - Series of different data types stored in a collection

String	Float	Integer	Boolean	List
Title	Rating	Times Viewed	Favorite	Seen By
Star Wars	9.8	13	True	Alice, Bob
Matrix	8.5	23	False	Charlie
Indiana Jones	6.1	3	False	Daniel, Evie

#Topic-4 Logical & Boolean Operators

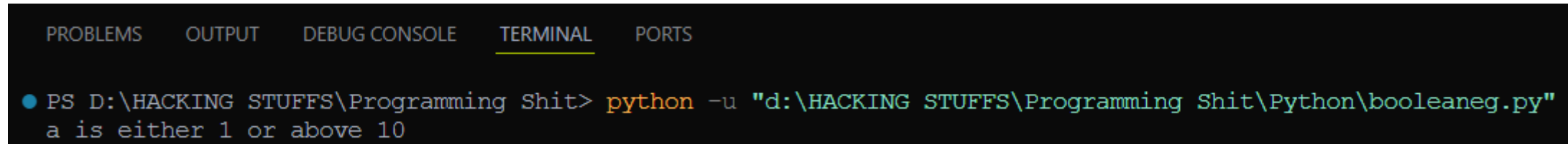
Logical operators allow assignment and comparisons to be made and are used in conditional testing (such as if statements).

Logical Operation	Operator	Example
Equivalence	==	if x == 5
Less than	<	if x < 5
Less than or equal to	<=	if x <= 5
Greater than	>	if x > 5
Greater than or equal to	>=	if x >= 5

- Code1

```
a = 1
if a == 1 or a > 10:
    print("a is either 1 or above 10")
```

- Output

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. Below the tabs, a command prompt shows the execution of a Python script: 'PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\booleaneg.py"'. The output of the script is 'a is either 1 or above 10'.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\booleaneg.py"
a is either 1 or above 10
```

- Code

```
name = "Jerry"

pentester = True

if name == "Jerry" and pentester == True:

    print ("Jerry is a Thwarted Pentester")

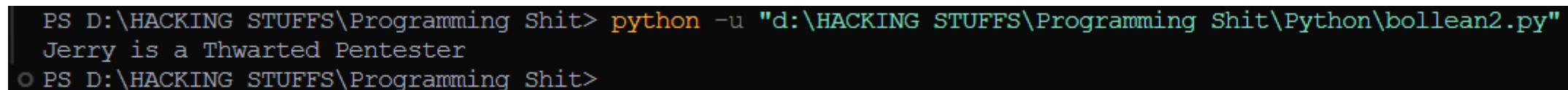
elif name == "Johnny" and pentester == True:

    print ("Identical Mistake")

else:

    print("Not sure who is Pentester!!!!")
```

- Output

A screenshot of a terminal window with a dark background. It shows the execution of a Python script: 'PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\bollean2.py"'. The output is 'Jerry is a Thwarted Pentester'. The prompt then changes to 'PS D:\HACKING STUFFS\Programming Shit>'.

```
PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\bollean2.py"
Jerry is a Thwarted Pentester
PS D:\HACKING STUFFS\Programming Shit>
```

#Topic-5 IF statements

Using "if statements" allows programs to make decisions. They let a program chose a decision based on a condition.

- Mini Project (Using IF elif Statement)

```
"""
```

```
In this project, you'll create a program that calculates the total  
cost of a customers shopping basket, including shipping.
```

- ```
- If a customer spends over $100, they get free shipping
- If a customer spends < $100, the shipping cost is $1.20 per kg of the baskets weight
```

```
Print the customers total basket cost (including shipping) to complete this exercise.
```

```
"""
```

```
shipping_cost_per_kg = 1.20
```

```
customer_basket_cost = 34
```

```
customer_basket_weight = 44
```

```
if (customer_basket_weight ≥ 100):

 print ('Free Shipping!!!')

else:

 shipping_cost = customer_basket_weight * shipping_cost_per_kg

 customer_basket_cost += shipping_cost

print("Total basket cost including shipping is " + str(customer_basket_cost))
```

- Output

```
PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\ifstatement.py"
Total basket cost including shipping is 86.8
PS D:\HACKING STUFFS\Programming Shit>
```

---

## #Topic-6 Loops

In programming, loops allow programs to iterate and perform actions a number of times. There are two types of loops, `for` and `while` loops.

### While Loop

We can have the loop run indefinitely or (similar to an if statement) determine how many times the loop should run based on a condition.

- Code

```
i = 0

while i ≤ 10:

 print(i)

 i = i + 1
```

- Output

```
python -u "d:\HACKING STUFFS\Programming Shit\Python\whileloop.py"
```

```
0
1
2
3
4
5
6
7
8
9
10
```

This while loop will run 10 times, outputting the value of the i variable each time it iterates (loops). Let's break this down:

- The i variable is set to 1
- The while statement specifies where the start of the loop should begin
- Every time it loops, it will start at the top (outputting the value of i)
- Then it goes to the next line in the loop, which increases the value of i by 1
- Then (as there is no more code for the program to execute), it goes to the top of the loop, starting the process over again
- The program will keep on looping until the value of the i variable is greater than 10

## For Loop

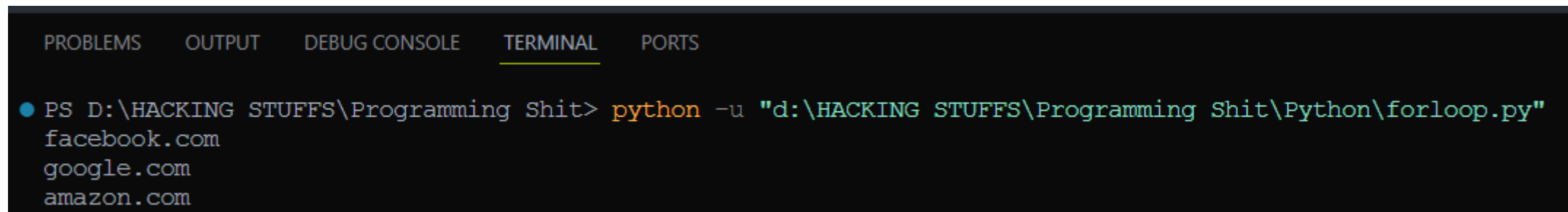
For loop is used to iterate over a sequence such as a list. Lists are used to store multiple items in a single variable, and are created using square brackets.



- Code

```
websites = ["facebook.com", "google.com", "amazon.com"]
for site in websites:
 print(site)
```

- Output



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\forloop.py"
facebook.com
google.com
amazon.com
```

This for loop shown in the code block above, will run 3 times, outputting each website in the list. Let's break this down:

- The list variable called websites is storing 3 elements
- The loop iterates through each element, printing out the element
- The program stops looping when it's been through each element in the loop

---

## #Topic-7 Function Statements

- As programs start to get bigger and more complex, some of your code will be repetitive, writing the same code to do the same calculations, and this is where functions come in. A function is a block of code that can be called at different places in your program.
- You could have a function to work out a calculation such as the distance between two points on a map or output formatted text based on certain conditions. Having functions removes repetitive code, as the function's purpose can be used multiple times throughout a program.
- Mini Project Creating Functions

```
"""
```

```
In this project, you'll create a program that that tells
you when the value of your Bitcoin falls below $30,000.
```

You will need to:

- Create a function to convert Bitcoin to USD
- If your Bitcoin falls below \$30,000 print a message.

You can assume that 1 Bitcoin is worth \$40,000

==> You've redeemed a hint. Replace the \_'s with code to complete this exercise.

```
"""
```

```
bitcoin_amount = 1.2
bitcoin_to_usd = 40000
```

```
1) write a function to calculate bitcoin to usd
def bitcoinToUSD(bitcoin_amount, bitcoin_value_usd):
 usd_value = bitcoin_amount * bitcoin_value_usd
 return usd_value
```

```
investment_in_usd = bitcoinToUSD(bitcoin_amount, bitcoin_to_usd)
if investment_in_usd <= 30000:
 print("Investment below $30,000! SELL!")
else:
 print("Investment above $30,000")
```

- Output

```
python -u "d:\HACKING STUFFS\Programming Shit\Python\function_miniProject.py"
Investment above $30,000
PS D:\HACKING STUFFS\Programming Shit>
```

In Python, you can read and write from files. We've seen that in cyber security, it's common to write a script and import or export it from a file; whether that be as a way to store the output of your script or to import a list of 100's of websites from a file to enumerate.

```
f = open("file_name", "r")
print(f.read())
```

To open the file, we use the built-in `open()` function, and the "r" parameter stands for "read" and is used as we're reading the contents of the file. The variable has a `read()` method for reading the contents of the file. You can also use the `readlines()` method and loop over each line in the file; useful if you have a list where each item is on a new line. In the example above, the file is in the same folder as the Python script; if it were elsewhere, **you would need to specify the full path of the file.**

You can also create and write files. If you're writing to an existing file, you open the file first and use the "a" in the open function after the filename call (which stands for append). If you're writing to a new file, you use "w" (write) instead of "a".

```
f = open("demofile1.txt", "a") # Append to an existing file
f.write("The file will include more text..")
f.close()

f = open("demofile2.txt", "w") # Creating and writing to a new file
f.write("demofile2 file created, with this content in!")
f.close()
```

---

## #Topic-9 Import

- In Python, we can import libraries, which are a collection of files that contain functions.
- Think of importing a library as importing functions you can use that have been already written for you.
- For example, there is a "date" library that gives you access to hundreds of different functions for anything date and time-related.

- Code

```
import datetime

current_time = datetime.datetime.now()

print(current_time)
```

- Output

```
PS D:\HACKING STUFFS\Programming Shit> python -u "d:\HACKING STUFFS\Programming Shit\Python\import.py"
2024-06-01 16:34:27.522222
PS D:\HACKING STUFFS\Programming Shit> █
```

- We import other libraries using the `import` keyword. Then in Python, we use that import's library name to reference its functions.
- In the example above, we import `datetime`, then access the `.now()` method by calling `library_name.method_name()`.
- Copy and paste the example above into the code editor.

Here are some popular libraries you may find useful in scripting as a pentester:

- Request - simple HTTP library.
- [Scapy](#) - send, sniff, dissect and forge network packets
- [Pwntools](#) - a CTF & exploit development library.

**NOTE:** Many of these libraries are already built into the programming language; however, libraries written by other programmers not already installed in your machine can be installed using an application called pip, which is Python's package manager.