**1. What are the two values of the Boolean data type? How do you write them?**

Boolean data type stores the information in the form of “TRUE” or “FALSE”.

Representation in python:

a=True

b=False

The expressions True and False without “” (Case sensitivity to be maintained) is interpreted by Python as Boolean. If we assign “True” or “False”, Python will interpret it as strings, not BOOLEAN.

**2. What are the three different types of Boolean operators?**

The three basic BOOLEAN operators are: AND, OR, NOT

**AND** expression takes two input conditional statements and outputs a BOOLEAN is terms of True or False.

* It returns True when both conditions are True. Even if one of the two is False, AND returns False

**OR** expression takes two input conditional statements and outputs a BOOLEAN is terms of True or False.

* It returns True when either of both conditions are True. Only if both the conditions are False, OR returns False

**NOT** expression takes one input conditional statements and outputs a BOOLEAN is terms of True or False.

* It returns True when the condition is False, It returns False when the condition is True
* It returns the opposite (NOT) of what the input condition implies

**3. Make a list of each Boolean operator's truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluates).**

The truth tables are filled in here in terms of True and False for ease of understanding. The truth tables can be represented in terms of numbers 0 and 1 also. Numerical 0 represents Boolean equivalent of False and numerical 1 represents Boolean equivalent of True.

Typecasting Boolean True to int(True) returns 1. Similarly int(False) returns 0

|  |  |
| --- | --- |
| **NOT Truth Table** | |
| Conditional Input 1 | Output |
| True | False |
| False | True |

|  |  |  |
| --- | --- | --- |
| **AND Truth Table** | | |
| Conditional Input 1 | Conditional Input 2 | Output |
| True | True | True |
| False | True | False |
| True | False | False |
| False | False | False |

|  |  |  |
| --- | --- | --- |
| **OR Truth Table** | | |
| Conditional Input 1 | Conditional Input 2 | Output |
| True | True | True |
| False | True | True |
| True | False | True |
| False | False | False |

**4. What are the values of the following expressions?**

(5 > 4) and (3 == 5): False

not (5 > 4): False

(5 > 4) or (3 == 5): True

not ((5 > 4) or (3 == 5) : False

(True and True) and (True == False): False

(not False) or (not True): True

**5. What are the six comparison operators?**

|  |  |
| --- | --- |
| **Comparison Operators** | |
| **==** | Equal To |
| **>** | Greater Than |
| **<** | Lesser Than |
| **>=** | Greater Than or Equal To |
| **<=** | Lesser Than or Equal To |
| **<>** | Not Equal To |

**6. How do you tell the difference between the equal to and assignment operators? Describe a condition and when you would use one.**

Assignment refers to assigning a value to a variable through an expression. For example, a=10 is an assignment, where the variable ‘a’ is assigned a value of 10 using the assignment operator (=). Whenever, ‘a’ gets called in the code, python will interpret it as 10.

On the contrary, Equal to (==) is a comparison operator. This takes two inputs and compares them against each other. If the expressions provided as inputs represent the same value, it returns True, else returns False. For example, the expression 1==2 returns False, but 1==(5/5)returns True

Sample Code using Assignment and Equal to:

#Checking if the value stored in the variable is equal to 5 or not

#Variable value assignment using assignment operator (=)

var=5

#if condition to compare if the variable ‘var’ has 5 in it or not using comparison operator (==)

if var==5:

print(‘Variable ‘var’ has the correct value’)

else:

print(‘Variable ‘var’ has the incorrect value’)

**The same code has been stored in GitHub repository in a Jupyter notebook format:**

[**https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python\_Basics/Python\_Basics\_Assignment\_2\_Question\_6.ipynb**](https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python_Basics/Python_Basics_Assignment_2_Question_6.ipynb)

**7. Identify the three blocks in this code:**

spam = 0

**if spam == 10:**

**print('eggs')**

**if spam > 5:**

**print('bacon')**

**else:**

**print('ham')**

print('spam')

print('spam')

**The three code blocks have been highlighted by changing the font colour to Purple (Block 1), Green (Block 2) and Brown (Block 3)**

**8. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! if anything else is stored in spam.**

spam=int(input(‘Enter the integer value to be stored in the variable spam : ‘))

if spam==1:

print(‘Hello’)

elif spam==2:

print(‘Howdy’)

else:

print(‘Greetings!’)

**The same code has been stored in GitHub repository in a Jupyter notebook format:**

[**https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python\_Basics/Python\_Basics\_Assignment\_2\_Question\_8.ipynb**](https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python_Basics/Python_Basics_Assignment_2_Question_8.ipynb)

**9. If your programme is stuck in an endless loop, what keys you’ll press?**

Ctrl+C

**10. How can you tell the difference between break and continue?**

break: This discontinues the current loop within which it has been called. ‘break’ exits the current loop and the code skips to the next line after the end of the current loop.

continue: The code returns back to first line of the current loop within which ‘continue’ is called. It skips the lines within the loop after the continue command is called.

**11. In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?**

**Syntax of range() – range(starting value, ending value, step size)**

**range() returns range of integers starting from the starting value till ending value – 1 stepped by the input step size provided.**

**Starting value and step size take default values of 0 and 1 respectively when they are not explicity mentioned.**

* range(10) : for loop iterates from starting 0 to ending 9 (which is mentioned value – 1)
* range(0,10) : for loop iterates from starting 0 to ending 9. Here the starting value is supplied as input into the range(). If the same function was defined as range(1,10), for loop would iterate from 1 to 9 rather than 0 to 9
* range(0,10,1): for loop iterates from starting 0 to ending 9. Here the starting value and step size are supplied as inputs into the range(). If the same function was defined as range(1,10,3), for loop would iterate from 1 to 9 rather stepping by 3 i.e 1, 4, 7 rather than the full range of 0 to 9

**12. Write a short program that prints the numbers 1 to 10 using a ‘for’ loop. Then write an equivalent program that prints the numbers 1 to 10 using a ‘while’ loop.**

#Using for loop

for i in range(1,11):

print(i)

#Using while loop

i=1

while i<11:

print(i)

i+=1

**The same code has been stored in GitHub repository in a Jupyter notebook format:**

[**https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python\_Basics/Python\_Basics\_Assignment\_2\_Question\_12.ipynb**](https://github.com/arvindhhp/iNeuronAssignments/blob/main/Python_Basics/Python_Basics_Assignment_2_Question_12.ipynb)

**13. If you had a function named bacon() inside a module named spam, how would you call it after importing spam?**

Syntax is module\_name.fun\_name()

For this example:

**spam.bacon()**