**Python Assignment 2**

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

The Boolean data type represents a binary value, indicating either true or false. In Python, the two values of the Boolean data type are written as True and False. These values are used to represent the truth or falsity of conditions or logical statements in your code.

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

Boolean operators are used to perform logical operations on Boolean values (True or False). There are three primary Boolean operators:

AND (and): This operator returns True if both operands are True, otherwise, it returns False.

Example:

True and False # Evaluates to False

OR (or): This operator returns True if at least one of the operands is True, otherwise, it returns False.

Example:

True or False # Evaluates to True

NOT (not): This operator is a unary operator that returns the opposite of the Boolean value. If the operand is True, not returns False, and if the operand is False, not returns True.

Example:

not True # Evaluates to False

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 evaluate ).

Here are the truth tables for the three primary Boolean operators: AND (and), OR (or), and NOT (not). In each table, T represents True, and F represents False.

AND (and) Truth Table:

Operand 1 Operand 2 Result

T T T

T F F

F T F

F F F

OR (or) Truth Table:

Operand 1 Operand 2 Result

T T T

T F T

F T T

F F F

NOT (not) Truth Table:

Operand Result

T F

F T

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?

The six comparison operators in Python are used to compare values and produce Boolean results (True or False) based on the comparison. Here are the six comparison operators:

Equal to (==): This operator checks if two values are equal.

Example: x == y returns True if x is equal to y.

Not equal to (!=): This operator checks if two values are not equal.

Example: x != y returns True if x is not equal to y.

Greater than (>): This operator checks if the left operand is greater than the right operand.

Example: x > y returns True if x is greater than y.

Less than (<): This operator checks if the left operand is less than the right operand.

Example: x < y returns True if x is less than y.

Greater than or equal to (>=): This operator checks if the left operand is greater than or equal to the right operand.

Example: x >= y returns True if x is greater than or equal to y.

Less than or equal to (<=): This operator checks if the left operand is less than or equal to the right operand.

Example: x <= y returns True if x is less than or equal to y.

6.How do you tell the difference between the equal to and assignment operators?Describe a

condition and when you would use one.

The equal to (==) operator and the assignment (=) operator serve different purposes in programming, and they are used in distinct contexts.

Equal To Operator (==):

The equal to operator (==) is used to compare two values to check if they are equal. It returns True if the values are equal and False otherwise.

For example:

x = 5

y = 10

result = x == y # This will be False because 5 is not equal to 10

Assignment Operator (=):

The assignment operator (=) is used to assign a value to a variable. It assigns the value on the right side of the operator to the variable on the left side.

For example:

x = 5 # Assigns the value 5 to the variable x

the equal to operator is used for comparison, and the assignment operator is used for assigning values to variables.

7. Identify the three blocks in this code:

spam = 0

if spam == 10: - 1st block

print('eggs')

if spam > 5: - 2nd block

print('bacon')

else: - 3rd block

print('ham')

print('spam')

print('spam')

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 = 2 # we can change the value of spam to test different cases

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

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

If your program is stuck in an endless loop and you want to interrupt its execution, you can typically press the keyboard combination Ctrl + C (Control key and the C key simultaneously). This keyboard shortcut is often used to send an interrupt signal to the running program, causing it to terminate.

Here's what you can do:

If you are running the program in a terminal or command prompt, click on the terminal window to make sure it's active.

Press Ctrl + C on your keyboard.

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

break and continue are two control flow statements in Python that are used within loops (for and while) to control the flow of the loop execution. They serve different purposes:

break Statement:

The break statement is used to immediately terminate the innermost loop (in which it is located) and continue with the next statement after the loop.

It is typically used when a certain condition is met, and you want to exit the loop early.

Once the break statement is encountered, the loop stops executing, and the program proceeds to the next statement after the loop.

Example:

for i in range(1, 6):

if i == 3:

break

print(i)

Output:

1

2

continue Statement:

The continue statement is used to skip the rest of the current iteration of the loop and move to the next iteration.

It is often used when you want to skip certain iterations based on a specific condition.

When the continue statement is encountered, the remaining code within the current iteration is skipped, and the loop proceeds to the next iteration.

Example:

for i in range(1, 6):

if i == 3:

continue

print(i)

Output:

1

2

4

5

break is used to exit the loop prematurely when a specific condition is met.

continue is used to skip the current iteration and move to the next iteration of the loop.

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

In a for loop, range(10), range(0, 10), and range(0, 10, 1) are different ways to define the range of values that the loop will iterate over. However, they all essentially achieve the same result, which is to iterate over the numbers from 0 to 9 (inclusive).

Here's the breakdown of each variant:

range(10):

This form of range specifies only the stop value. It starts from 0 (implicitly) and stops at 10 (exclusive). The step size is implicitly 1.

for i in range(10):

print(i)

Output:

0

1

2

3

4

5

6

7

8

9

range(0, 10):

This form of range specifies both the start and stop values. It starts from 0 (explicitly) and stops at 10 (exclusive). The step size is implicitly 1.

for i in range(0, 10):

print(i)

Output is the same as the previous example.

range(0, 10, 1):

This form of range specifies the start, stop, and step values. It starts from 0 (explicitly), stops at 10 (exclusive), and increments by 1.

for i in range(0, 10, 1):

print(i)

Output is the same as the previous examples.

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.

Here's the short program that prints the numbers 1 to 10 using a for loop:

for i in range(1, 11):

print(i)

And here's the equivalent program that prints the numbers 1 to 10 using a while loop:

i = 1

while i <= 10:

print(i)

i += 1

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

importing spam?

If we have a function named bacon() inside a module named spam, and we want to call this function after importing the spam module, we would use the following syntax:

import spam

spam.bacon()