Assignment 2

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

Ans . The Boolean data type has two values: true and false. In programming languages, these values are typically represented using the keywords "true" and "false." For example, in languages like Python, JavaScript, or Java, you would write:

Example :

# Boolean values

my\_boolean\_variable = True

another\_boolean\_variable = False

# Print the values

print("my\_boolean\_variable:", my\_boolean\_variable)

print("another\_boolean\_variable:", another\_boolean\_variable)

# Logical operations

result\_and = my\_boolean\_variable and another\_boolean\_variable

result\_or = my\_boolean\_variable or another\_boolean\_variable

result\_not = not my\_boolean\_variable

# Print the results of logical operations

print("result\_and:", result\_and)

print("result\_or:", result\_or)

print("result\_not:", result\_not)

The output of above python script will be as follows:

my\_boolean\_variable: True

another\_boolean\_variable: False

result\_and: False

result\_or: True

result\_not: False

This output demonstrates the use of Boolean variables and some basic logical operations (AND, OR, NOT) with them.

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

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

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

result = True and False # Result is False

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

result = True or False # Result is True

NOT Operator (not): This operator returns the opposite of the operand. If the operand is True, it returns False, and if the operand is False, it returns True.

result = not True  # Result is False

These Boolean operators are commonly used in conditional statements and logical expressions to control the flow of a program based on certain conditions.

**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 ).**

Ans. Following are the truth tables for each Boolean operator:

AND Operator (and):

|  |  |  |
| --- | --- | --- |
| Operand 1 | Operand 2 | Result |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

OR Operator (or):

|  |  |  |
| --- | --- | --- |
| Operand 1 | Operand 2 | Result |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

NOT Operator (not):

|  |  |
| --- | --- |
| Operand | Result |
| True | False |
| False | True |

These truth tables represent the outcome of each Boolean operation based on all possible combinations of Boolean values for the operands.

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

1.(5 > 4) and (3 == 5):

The first part, 5 > 4, is True.

The second part, 3 == 5, is False.

The result of the entire expression is True and False, which is False.

2.not (5 > 4):

The inner expression, 5 > 4, is True.

The not operator negates it, so the result is False.

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

The first part, 5 > 4, is True.

The second part, 3 == 5, is False.

The result of the entire expression is True or False, which is True.

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

The inner expression, (5 > 4) or (3 == 5), is True.

The not operator negates it, so the result is False.

5. (True and True) and (True == False):

The first part, True and True, is True.

The second part, True == False, is False.

The result of the entire expression is True and False, which is False.

6.(not False) or (not True):

The first part, not False, is True.

The second part, not True, is False.

The result of the entire expression is True or False, which is True.

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

Ans. The six comparison operators are used to compare values and return a Boolean result (True or False). Here they are:

Equal to (==): Checks if two values are equal.

5 == 5  # True

Not equal to (!=): Checks if two values are not equal.

5 != 3  # True

Greater than (>): Checks if the left operand is greater than the right operand.

5 > 3  # True

Less than (<): Checks if the left operand is less than the right operand.

3 < 5  # True

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

5 >= 5  # True

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

3 <= 5  # True

These operators are commonly used in conditional statements and comparisons within programming to control the flow of the program based on certain conditions.

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

Ans. The equal to operator (==) and the assignment operator (=) serve different purposes in programming.

Equal to operator (==):

Purpose: The equal to operator is a comparison operator used to check if two values are equal.

Example:

x = 10

y = 5

if x == y:

   print("x is equal to y")

else:

   print("x is not equal to y")

Condition: The equal to operator is used when you want to compare two values or variables to determine if they are equal. It is commonly used in conditional statements, such as if statements, to make decisions based on whether a condition is true or false.

2. Assignment operator (=):

Purpose: The assignment operator is used to assign a value to a variable.

Example:

x = 10

Condition: The assignment operator is used whenever you need to give a variable a specific value. It is not used for comparison. Instead, it assigns the value on the right side of the operator to the variable on the left side.

**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')

Ans.

Block 1:

if spam == 10:

   print('eggs')

This block contains the print('eggs') statement and will be executed if the condition spam == 10 is true.

Block 2:

if spam > 5:

   print('bacon')

This block contains the print('bacon') statement and will be executed if the condition spam > 5 is true.

Block 3:

else:

   print('ham')

This block is associated with the if spam > 5 statement. It contains the print('ham') statement and will be executed if the condition spam > 5 is false.

The print('spam') statements at the end are not part of any specific block as they are not indented under an if statement.

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.

Ans. Python code that meets the specified conditions is as follows:

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

if spam == 1:

  print('Hello')

elif spam == 2:

  print('Howdy')

else:

  print('Greetings!')

In this code:

If the value stored in spam is equal to 1, it will print 'Hello'.

If the value is equal to 2, it will print 'Howdy'.

If the value is anything other than 1 or 2, it will print 'Greetings!'.

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

Ans . If our program is stuck in an endless loop and you need to interrupt or stop it, you can typically press the "Ctrl" (Control) key along with the "C" key on your keyboard. This keyboard shortcut is used to send an interrupt signal to the running program in many operating systems, including Windows, Linux, and macOS.

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

In Python, break and continue are both control flow statements used within loops (such as for and while loops), but they serve different purposes.

break statement:

When a break statement is encountered within a loop, it immediately terminates the entire loop, and the program continues with the next statement outside the loop.

It is often used when a specific condition is met, and you want to exit the loop prematurely.

Example :

for i in range(5):

if i == 3:

break

print(i)

Output :

0

1

2

continue statement:

When a continue statement is encountered within a loop, it skips the rest of the code inside the loop for the current iteration and proceeds to the next iteration.

It is typically used when you want to skip certain iterations based on a condition, but continue with the next ones.

Example :

for i in range(5):

if i == 2:

continue

print(i)

Output :

0

1

3

4

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

range(10):

This creates a sequence of numbers from 0 up to (but not including) 10.

It is equivalent to range(0, 10, 1) with a default step of 1.

range(0, 10):

This also creates a sequence of numbers from 0 up to (but not including) 10.

It explicitly specifies the starting point (0) and assumes a default step of 1.

range(0, 10, 1):

This also creates a sequence of numbers from 0 up to (but not including) 10.

It explicitly specifies the starting point (0), ending point (10), and step size (1). However, specifying a step size of 1 is redundant because it is the default.

**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.**

Ans. # Using a for loop

for i in range(1, 11):

print(i)

# Using a while loop

counter = 1

while counter <= 10:

print(counter)

counter += 1

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

import spam

spam.bacon()

This assumes that the bacon() function is defined within the spam module. The import spam statement imports the entire spam module, and then you can use dot notation (spam.bacon()) to access and call the bacon() function within the module.