INEURON ASSIGNMENT 2

Python

kaushik

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

Ans : The two values of the Boolean data type are **True(means 1)** and **False(means 0)**.

```
# how to write the boolean values in python

bool_a=True

print ("the true data types are", bool_a ,'and its class is ',type(bool_a) , 'and its numeric values', int(bool_a))

bool_b=False

print ("the true data types are", bool_b ,'and its class is ',type(bool_b),'and its numeric values', int(bool_b))

the true data types are True and its class is  <class 'bool'> and its numeric values 1

the true data types are False and its class is  <class 'bool'> and its numeric values 0
```

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

Ans : The three different types of Boolean operators are given below

Comparison Operator

Equal to (operator type is comparison & operator sign is ==)

Example :

```
#comparison operator

x = 5

y = 8

print("x == y:", x == y)

x == y: False
```

Logical Operator

and (operator type is logical & operator sign is and)

True if both are true

Example :

```
#Logical Operators

print((9 > 7) and (2 < 4))

True
```

Logical Operator

or (operator type is logical & operator sign is or)

True if at least one is true.

Example:

```
#Logical Operators

print ((8 == 8) or (6!= 6))

True
```

Logical Operator

not (operator type is logical & operator sign is not)

True only if false

```
#Logical Operators

print(not(3 <= 1))

True
```

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 :

== Truth Table

X	*==*	*Y*	*Returns*
True (1)	==	**True (1)**	**True(1)**
True(1)	==	**False(0)**	**False(0)**
False(0)	==	**True(1)**	**False(0)**
False(0)	==	**False(0)**	**True(1)**

AND Truth Table

X	*and*	*Y*	*Returns*
True (1)	**and**	**True (1)**	**True(1)**
True(1)	**and**	**False(0)**	**False(0)**
False(0)	**and**	**True(1)**	**False(0)**
False(0)	**and**	**False(0)**	**False(0)**

OR Truth Table

X	*or*	*Y*	*Returns*
True (1)	**or**	**True (1)**	**True(1)**
True(1)	**or**	**False(0)**	**True(1)**
False(0)	**or**	**True(1)**	**True(1)**
False(0)	**or**	**False(0)**	**False(0)**

NOT Truth Table

not	*X*	*Returns*
Not	**True (1)**	**False (0)**
Not	**False (0)**	**True (1)**

4. What are the values of the following expressions?

Ans :

(5 > 4) and (3 == 5) = expression is False.

not (5 > 4) = expression is False.

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

not ((5 > 4) or (3 == 5)) = expression is False.

(True and True) and (True == False) = expression is False.

(not False) or (not True) = expression is True.

5. What are the six comparison operators?

Ans : The six comparison operators are given below.

Operator Name	Meaning	Example
==	Equal to	x = 5 y = 8 print("x == y:", x == y) x == y: False
!=	Not Equal to	x = 5 y = 8 print("x != y:", x != y) x != y: True
<	Less than	x = 5 y = 8 print("x < y:", x < y) x < y: True
>	Greater than	x = 5 y = 8 print ("x > y:", x > y) x > y: False
<=	Less than or equal to	x = 5 y = 8 print("x <= y:", x <= y) x <= y: True

>=	Greater than or equal to	x = 5 y = 8 print("x >= y:", x >= y) x >= y: False

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

Ans :

= sign indicates assignment operator where == sign indicates equal to operator.

= assign values from right side operands to the left side operand

== checks if the value of two operands are equal or not equal.

One use case is given below:

```
# assignment operator
# in assignment operator i just assign the value of a
a = 5
print ("Print the value of a", a)

# equal to operator
# equal to operator checks the both variable(operands) values and
# on that basis it can print the result with the if else conditions
a = int(input("Enter the value of a"))
b = int(input("Enter the value of b"))
if a == b:
   print("Value are matched")
else :
   print("Value are not matched")
```

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 :

```
spam = 0
if spam == 10:
    print('eggs')        # indent increased, block A
    if spam > 5:         # block A
        print('bacon')   # still block A, indent increased, block B inside block A
    else:                # still block A, indent decreased, block B ended in line above
        print('ham')     # still block A, indent increased, block C inside block A
    print('spam')        # still block A, indent decreased, block C ended in line above
print('spam')            # indent decreased, block A ended in line above
```

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:

```
# different different condition different prints as in output
#first spam initialization
spam=0;
#if value 1 is stored in spam then prints Hello
if spam==1:
    print("Hello")
#if 2 is stored in spam then prints Howdy
elif spam==2:
    print("Howdy")
# if anything stored is spam then prints Greetings
else :
```

print("Greetings!")

Screenshot :

```
# different differnt condition different prints as in output
#first spam initialization
spam=0;
#if value 1 is stored in spam then prints Hello
if spam==1:
    print("Hello")
#if 2 is stored in spam then prints Howdy
elif spam==2:
    print("Howdy")
# if anything stored is spam then prints Greetings
else :
    print("Greetings!")
```

Now test with different case

Case 1 if spam is 1:

```
# different differnt condition different prints as in output
#first spam initialization
spam=1;
#if value 1 is stored in spam then prints Hello
if spam==1:
    print("Hello")
#if 2 is stored in spam then prints Howdy
elif spam==2:
    print("Howdy")
# if anything stored is spam then prints Greetings
else :
    print("Greetings!")
```

Output “Hello”

Case 2 if spam is 2:

```
# different differnt condition different prints as in output
#first spam initialization
spam=2;
#if value 1 is stored in spam then prints Hello
if spam==1:
    print("Hello")
#if 2 is stored in spam then prints Howdy
elif spam==2:
    print("Howdy")
# if anything stored is spam then prints Greetings
else :
    print("Greetings!")
```

Output “Howdy”

Case 3 if spam is ineuron

```python
# different differnt condition different prints as in output
#first spam initialization
spam="ineuron";
#if value 1 is stored in spam then prints Hello
if spam==1:
    print("Hello")
#if 2 is stored in spam then prints Howdy
elif spam==2:
    print("Howdy")
# if anything stored is spam then prints Greetings
else :
    print("Greetings!")
```

Output “Greetings!”

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

Ans : to stop an infinite loop we can press : ctrl + c

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

Ans : in simple words we can say that break statement stops the entire process of the loop where continue statement only stops the current iteration of the loop.

Screenshot for break statement :

```python
#example of break statement
for kaushik in range(0,5):
    print(f'Outer For Loop Iteration: {kaushik}')
    for ineuron in range(0,10):
        if ineuron == 5:
            break
        print(f'--Inner For Loop Iteration: {ineuron}')
```

Screenshot for another break statement :

```python
#break statement
for chance in range(1,4):
    passw = input("\nEnter a password:")
    if passw == 'python':
       print ("--correct password\n --- unlocking your system")
       break
    print(f"incorrect attemt: {chance}")
```

```
Enter a password: python
--correct password
 --- unlocking your system
```

Screenshot for continue statement:

```python
#example of continue statement
for kaushik in range(0,8):
    if kaushik == 5:
        continue
    print(f'Iteration: {kaushik}')
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 6
Iteration: 7
```

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

Ans : there is no difference between above conditions, all are printed upto range 10 (from 0 to 9)

There different conditions screenshot is given below:

Case 1 range(10):

```
#range10
for kaushik1 in range(10):
    print(f'itr: {kaushik1}')
```

```
itr: 0
itr: 1
itr: 2
itr: 3
itr: 4
itr: 5
itr: 6
itr: 7
itr: 8
itr: 9
```

Case 2 range(0,10):

```
#range(0,10)
for kaushik1 in range(0,10):
    print(f'itr: {kaushik1}')
```

```
itr: 0
itr: 1
itr: 2
itr: 3
itr: 4
itr: 5
itr: 6
itr: 7
itr: 8
itr: 9
```

Case 3 range(0,10,1):

```
#range(0,10,1)
for kaushik1 in range(0,10,1):
    print(f'itr: {kaushik1}')
```

```
itr: 0
itr: 1
itr: 2
itr: 3
itr: 4
itr: 5
itr: 6
itr: 7
itr: 8
itr: 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.

Ans:

Prints the numbers 1 to 10 using a for loop.

Code:

```
#prints the numbers 1 to 10 using a for loop
for i in range(1, 11):
    print(i)
```

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

Prints the numbers 1 to 10 using while loop.

while_loop_number_print.py

module_python > while_loop_number_print.py > ...

```
#prints the numbers 1 to 10 using while loop
# init the value i =1
i = 1
while(i<=10):
    print(f"the number is {i} in its {i} loop.")
    i += 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE Python

```
the number is 1 in its 1 loop.
the number is 2 in its 2 loop.
the number is 3 in its 3 loop.
the number is 4 in its 4 loop.
the number is 5 in its 5 loop.
the number is 6 in its 6 loop.
the number is 7 in its 7 loop.
the number is 8 in its 8 loop.
the number is 9 in its 9 loop.
the number is 10 in its 10 loop.
PS D:\Big_data\cloud_hands_on\Ineuron_assignments\python>
```

Ln 4, Col 14 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit Go Live

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

Ans :

Inside spam.py

```python
# creating bacon func with parameter
def bacon(name):
  print(f"welcome to python world, {name}")
```

Inside call.py

```python
# calling spam module inside call.py file
# And from that spam module calling the func bacon with params
import spam
spam.bacon("kaushik")
```

out put :

welcome to python world, kaushik!

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    AZURE

PS D:\Big_data\cloud_hands_on\Ineuron_assignments\python> & "C:/Program Files/Pyt
hon311/python.exe" d:/Big_data/cloud_hands_on/Ineuron_assignments/python/python_c
ustom_module/call.py
welcome to python world, kaushik
PS D:\Big_data\cloud_hands_on\Ineuron_assignments\python>

Python
Python
```