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

Ans:- The Boolean data type has two possible values: "true" and "false". These values represent the two logical states of True" and "false".

a = True b = False

print(type(a)) print(type(b))

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

```
 Ans:-
        The three different types of Boolean operators are:

AND Operator (and): This operator returns True if both operands are True, otherwise,
it returns False. It requires both conditions to be true for the entire expression to
be true.

OR Operator (or): This operator returns True if at least one of the operands is True.
It only requires one of the conditions to be true for the entire expression to be
true.

NOT Operator (not): This operator is used to negate the value of a Boolean expression.
It returns True if the expression is False, and False if the expression is 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)

In [ ]:

```
Ans:-
  ## AND Operator (and):
True and True -> True
True and False -> False
False and True -> False
False and False -> False

 ##  OR Operator (or):
True or True -> True
True or False -> True
False or True -> True
False or False -> False

 ## NOT Operator (not):
not True -> False
not False -> True
```

4. What are the values of the following expressions?

(5 > 4) and (3 == 5) not (5 > 4) (5 > 4) or (3 == 5) not ((5 > 4) or (3 == 5)) (True and True) and (True == False) (not False) or (not True)

In [11]:

```
print((5 > 4) and (3 == 5))
print(not (5 > 4))
print(5 > 4) or (3 == 5)
print(not (5 > 4) or (3 == 5))
print((True and True) and (True == False))
print(not False) or (not True)
```

```
False
False
True
False
False
True
```

Out[11]:

```
False
```

5. What are the six comparison operators?

In [ ]:

```
Ans:-
Equal to (==):

Not equal to (!=):

Greater than (>):

Less than (<):

Greater than or equal to (>=):

Less than or 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.

```
Ans:- # Equal To (==) Operator:
The equal to operator is used for comparison. It checks if the values on the left and
right sides are equal. If the values are the same, the expression evaluates to True;
otherwise, it evaluates to False.
For Example:-
x = 5
y = 10
result = x == y
print(result)
#Output: False

 Assignment (=) Operator:
The assignment operator is used to assign a value to a variable. It takes the value on
the right side and assigns it to the variable on the left side.
For Example:-
x = 5
 In this example, the value 5 is assigned to the variable x.
```

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:-
    In Python,code block refers to a collection of code that is in the same block or
indent. This is most commonly found in classes, functions and loops.
```

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

In [12]:

```
def spem_code(spam):
    if spam == 1:
        print("Hello")
    elif spam == 2:
        print("Howdy")
    else:
        print("Greetings")

spem_code(1)
spem_code(2)
spem_code(3)
```

```
Hello
Howdy
Greetings
```

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

```
Ans:-
    Pressing Ctrl and C keys simultaneously (or Command and . keys on macOS) will send
an interrupt signal to the running program. This will usually force the program to
terminate and break out of the endless loop.
```

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

```
Ans:-
    Break:
The break statement is used to terminate the loop immediately when a certain condition
is met.
When the break statement is encountered within a loop (such as for or while), the loop
is immediately exited, and the program continues with the next statement after the
loop.
It is often used to exit a loop prematurely when a specific condition is satisfied, or
when there is no need to continue further iterations.

For Example:-
```

In [13]:

```
for i in range(1, 6):
    if i == 4:
        break
    print(i)
```

```
1
2
3
```

```
  Continue:
        The continue statement is used to skip the current iteration of the loop and
move to the next iteration.
When the continue statement is encountered within a loop, the remaining code for that
iteration is skipped, and the loop continues with the next iteration.
It is often used when you want to skip certain elements or operations within a loop
based on some condition but continue with the next iterations.

For Example:-
```

In [14]:

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

```
1
2
4
5
```

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

```
Ans:-
#range(10):
When you use range(10) in a for loop, it will generate a sequence of numbers starting
from 0 (inclusive) up to 10 (exclusive), with a default step size of 1.
It will iterate over the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

For Example:-
```

In [18]:

```
for i in range(10):
    print(i)
```

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

```
 range(0, 10):
When you use range(0, 10) in a for loop, it will generate a sequence of numbers
starting from 0 (inclusive) up to 10 (exclusive), just like the previous example. The
range explicitly starts from 0 and goes up to 10.
The loop will still iterate over the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

For Example:-
```

In [19]:

```
for i in range(0, 10):
    print(i)
```

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

```
 range(0, 10, 1):
When you use range(0, 10, 1) in a for loop, it will also generate the same sequence of
numbers as the previous examples. The range starts from 0 and goes up to 10, with a
step size of 1. The step size of 1 means that it will increment by 1 for each
iteration.
The loop will still iterate over the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

For example:-
```

In [20]:

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

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

In [ ]:

```
Ans:-
    # For Loop
    # While Loop
```

In [21]:

```
## For loop
for i in range(1, 11):
    print(i)
```

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

In [22]:

```
## While loop
num = 1
while num <= 10:
    print(num)
    num += 1
```

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

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

In [ ]:

```
Ans:-
    After importing the module spam, you can call the function bacon()
```