

Assignment 2

Solution 1.

“True” and **“False”** are the two values of the Boolean data type.

We can write them as:-

```
x = True  
y = False
```

In this, the variable x is assigned the value **True**, and the variable y is assigned the value **False**.

Solution 2.

The three different types of Boolean operators are :- **AND**, **OR**, and **NOT**.

Solution 3.

List of each Boolean operator's truth tables.

AND operator

| Operand 1 | Operand 2 | Result |
|-----------|-----------|--------|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

Or operator

| Operand 1 | Operand 2 | Result |
|-----------|-----------|--------|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

NOT operator

| Operand | Result |
|---------|--------|
| true | false |
| false | true |

Solution 4.

```
[1]: (5 > 4) and (3==4)
```

```
[1]: False
```

```
[3]: not (5 > 4)
```

```
[3]: False
```

```
[4]: (5 > 4) or (3 == 5)
```

```
[4]: True
```

```
[5]: not ((5 > 4) or (3 == 5))
```

```
[5]: False
```

```
[6]: (True and True) and (True == False)
```

```
[6]: False
```

```
[7]: (not False) or (not True)
```

```
[7]: True
```

```
[ ]: |
```

Solution 5.

The six comparison operators are:-

==

!=

>

<

>=

<=

Solution 6.

Equal to operator (==): The equal to operator is used to compare whether two values are equal. It checks if the values on both sides of the operator are the same and returns a Boolean value of either true or false.

Example -

```
[8]: x = 6
     y = 9
     print(x==y)

False

[ ]: |
```

In this example, the equal to operator ($x == y$) compares the values of x and y . Since x is not equal to y , the result is False.

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

Example -

```
x = 5
y = x
```

In this, the assignment operator ($=$) assigns the value of x to the variable y . After the assignment, the value of y becomes 5.

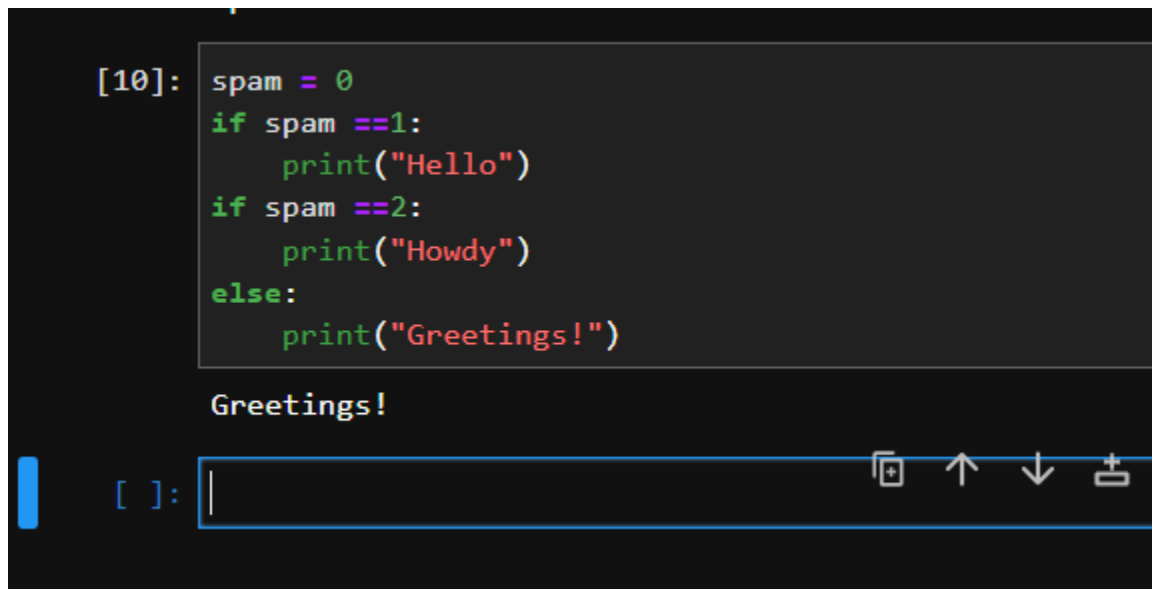
Solution 7.

The three blocks are everything inside the if statement and the lines `print('ham')`, `print('spam')` and `print('spam')`

Solution 8.

```
[10]: spam = 0
      if spam ==1:
          print("Hello")
      if spam ==2:
          print("Howdy")
      else:
          print("Greetings!")

Greetings!
```



Solution 9.

If your programme is stuck in an endless loop, then I'll press **Ctrl + C** (on Windows or Linux) or **Command +** (on macOS)

Solution 10.

The “*break statement*” is used to terminate the execution of the current loop. Whenever there is a need to end the loop, you need to add the break statement. Once the break statement is met, all the iterations are stopped, and the control is shifted outside the loop.

Example -

```
[1]: for i in range(1, 10):  
      if i == 5:  
          break  
      print(i)  
  
1  
2  
3  
4  
  
[ ]: |
```

The “*continue statement*” is used to move to the next iteration, it is also used for termination, but unlike break, it is not used to terminate the entire execution of the loop but only the current iteration of the loop.

Example -

```
[3]: for i in range(1, 10):  
      if i == 6:  
          continue  
      print(i)
```

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

```
[ ]:
```

Solution 11.

In a for loop, the expressions **range(10)**, **range(0, 10)**, and **range(0, 10, 1)** are functionally equivalent and produce the same sequence of numbers.

The differences lie in the syntax and the arguments provided to the **range()** function.

The **range()** function in Python generates a sequence of numbers that can be used to iterate over in a loop. It takes up to three arguments: **start**, **stop**, and **step**.

Solution 12.

Program that prints the numbers 1 to 10 using a **for loop** and **while loop**.

```
[ ]: # using for loop
     for i in range(1,11):
         print(i)
```

```
[ ]: # using while loop
     i = 1
     while i<=10:
         print(i)
         i += 1
```


Solution 13.

```
[ ]: import spam  
spam.bacon()
```