

Assignment_2

1. What are the Boolean data type's two values? How do you go about writing them?

Ans.

The Boolean data type has two values in Python. These values are **True** and **False**.

We can initialize a Boolean variable as follows:

```
a = True
b = False
```

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

Ans.



The three different Boolean operators are **AND**, **OR** and **NOT**.

- **AND** - This operator results in True iff all the parts of the condition are True.
- **OR** - This operator results in True if at least one of the parts of the condition is True.
- **NOT** - This operator negates the condition.

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 evaluates to).

Ans.


AND Table

<u>Aa</u> Boolean Var 1	 Boolean Var 2	 Result
<u>True</u>	True	True
<u>True</u>	False	False
<u>False</u>	True	False
<u>False</u>	False	False

OR Table

<u>Aa</u> Boolean Var 1	 Boolean Var 2	 Result
<u>True</u>	True	True
<u>True</u>	False	True
<u>False</u>	True	True
<u>False</u>	False	False

NOT Table

<u>Aa</u> Boolean Var	 Result
<u>True</u>	False
<u>False</u>	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)

Ans.

- (5 > 4) and (3 == 5) evaluates to **False**, as one of the conditions is false.
- not (5 > 4) evaluates to **False**, as 'not' negates the result.

- $(5 > 4)$ or $(3 == 5)$ evaluates to **True**, as at least one of the conditions is true.
- $\text{not } ((5 > 4) \text{ or } (3 == 5))$ evaluates to **False**, as 'not' negates the result.
- (True and True) and $(\text{True} == \text{False})$ evaluates to **False**, as one of the conditions is false.
- (not False) or (not True) evaluates to **True**, as one of the conditions is true.

5. What are the six different types of reference operators?

Ans.

There are 6 different types of operators in Python :

- Arithmetic Operators
 1. +
 2. -
 3. /
 4. *
 5. **
 6. //
 7. %
- Assignment Operators
 1. =
 2. +=
 3. -=
 4. *=
 5. /=
 6. **=
 7. //=
 8. %=

- Relational Operators

1. ==

2. <

3. >

4. ≤

5. ≥

6. ≠

- Boolean Operators

1. AND

2. OR

3. NOT

- Conditional Operators

1. if else

- Bitwise Operators

1. ^

2. &

3. |

4. <<

5. >>

6. ~

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

Ans.

The equal to operator in Python is '==' and the assignment operator in Python is '='.

7. Describe a condition and when you would use one.

Ans.

A condition statement can be used to confirm whether something is true or false and proceed with the program accordingly. For example, a condition can be used to check whether a person is above 18 years old in order to apply for a driving license.

This is shown in the following piece of code.

```
...
if(age < 18): #condition
    print("Kid you are not eligible")
else:
    print("You are eligible.")
...
```

8. Recognize the following 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.

The blocks of code have been identified in the following snippet :

```
spam = 0          #first block
if spam == 10:
    print('eggs') #second block
    if spam > 5:
        print('bacon') #third block
    else:
        print('ham')
    print('spam')
print('spam')
```

9. Create a program that prints. If 1 is stored in spam, prints Hello; if 2 is stored in spam, prints Howdy; and if 3 is stored in spam, prints Salutations! if there's something else in spam.

Ans.

The following program does the job.

```
spam = int(input("Enter the value: "))

if spam == 1:
    print("Hello")

elif spam == 2:
    print("Howdy")

elif spam == 3:
    print("Salutations")

else:
    print("The print statement is not mentioned in the question!!")
```

10. If your program is stuck in an endless loop, what keys can you press?

Ans.

If our program is stuck in an infinite loop, we can try and press ctrl + c to break the loop.

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

Ans.

Break : The break statement in Python terminates the current loop and resumes execution at the next statement. For example

```
for i in range(10):
    if i == 5: break
    print(i)
```

```
# this code prints numbers 0 to 4, as soon as i == 5, the loop breaks and not  
# continued further.
```

Continue : The continue statement in Python returns the control to the beginning of the loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. For example

```
for i in range(10):  
    if i == 5: continue  
    print(i)  
  
# this code prints numbers 0 to 9 but does not print 5.  
# when i == 5, it takes the control to beginning of the loop  
# and skips the print statement for that iteration.
```

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

Ans.

First we see the range() function in Python.

range(start, end, skip)

Here, start - starting digit of the iterable.

end - ending digit of the iterable. (it stops at end-1)

skip - the increment between each digit.

In a for loop in Python,

- range(10) - This creates a range object which is basically an iterable. We can iterate through the numbers 0 to 9 using this.
- range(0, 10) - This also creates a range object, but here we explicitly mention the starting number of the range we want to iterate through. By default, it starts from 0. So here also we can iterate from 0 through 9.
- range(0, 10, 1) - This again creates a range object, the start is also mentioned here, and the last argument is the skip. Here, it is 1. This

mean while iterating, the increment is 1 digit. The skip argument is by default 1. Hence, this is also the same range object as the above two.

13. Using a for loop, write a short program that prints the numbers 1 to 10. Then, using a while loop, create an identical program that prints the numbers 1 to 10.

Ans.

The following snippet of code does the job.

```
# using for loop
for number in range(1,11):
    print(number)

# using while loop
number = 1
while number <= 11:
    print(number)
    number += 1
```

14. If you had a bacon() function within a spam module, what would you call it after importing spam?

Ans.

When we have imported spam module, we need to call the bacon function as

```
import spam
spam.bacon()
```