Name: SAPTARSHI SANYAL

EMAIL ID: saptarshi.edp@gmail.com

Solutions to Assignment\_2

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

Ans.

Two values of Boolean data types are **True** and **False.**

Boolean data type values could be assigned to variables in following way:

X = True

Y= False

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

Ans.

The three types of Boolean operators:

1. *The AND operator (&& or "and")*
2. *The OR operator (|| or "or")*
3. *The NOT operator (not)*
4. **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:

**Logical AND Truth table:**

|  |  |  |
| --- | --- | --- |
| Operator A | Operator B | Logical **AND** result |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

**Logical OR Truth table:**

|  |  |  |
| --- | --- | --- |
| Operator A | Operator B | Logical **OR** result |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

**Logical NOT Truth table:**

Logical **NOT** operator works with the single boolean value and returns the value as True if the boolean value is False and vice-versa (that is the opposite of it).

|  |  |
| --- | --- |
| Operator A | Logical **NOT** result |
| True | False |
| False | True |

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

Ans:

* 1. (5 > 4) and (3 == 5) 🡪 **False**
  2. not (5 > 4)🡪 **False**
  3. (5 > 4) or (3 == 5) 🡪 **True**
  4. not ((5 > 4) or (3 == 5)) 🡪 **False**
  5. (True and True) and (True == False) 🡪**False**
  6. (not False) or (not True)🡪**True**

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

Ans:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Syntax** |
| == | Equal to: True if both operands are equal | a == b |
| < | Less than: True if the left operand is less than the right | a < b |
| > | Greater than: True if the left operand is greater than the right | a > b |
| != or <> | Not equal to – True if operands are not equal | a != b  or  a <> b |
| >= | Greater than or equal to: True if left operand is greater than or equal to the right | a >= b |
| <= | Less than or equal to: True if left operand is less than or equal to the right | a <= b |

1. **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 checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false.

The assignment (“=”) operator is used to assign the value on the right to the variable on the left.

In the following case equal to operator checks whether integer values 5 is equal to 5:

5 == 5

It returns true.

In the following case an integer value is assigned into the variable a through assignment operator:

a = 10;

print(a)

This will print the value of variable a as 10 which was assigned to it in the preceding step.

1. **Identify the three blocks in this code:**

spam = 0

if spam == 10:

print(‘eggs’)---------------🡪Code Block 1

if spam > 5:

print(‘bacon’) ---------------🡪Code Block 2

else:

print(‘ham’)

print(‘spam’) -------🡪Code Block 3 print(‘spam’)

(\*\*) I had some doubts regarding this question in its indentation. I got revised question from Mr Deepranjan Gupta over mail.

1. **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:

spam =1

if (spam==1):

print('Hello')

elif (spam==2):

print('Howdy')

else:

print('Greetings!')

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

ANS: CTRL + C

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

ANS:

The break statement brings the execution flow of the program out of the loop but the continue statement brings the execution flow of the program back to the start of loop and re evaluates the loop’s condition.

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

Ans:

In a for loop, range() function can be called with three arguments. The first two arguments will be the start and stop values, and the third will be the step argument. The step is the amount that the variable is increased by after each iteration.

range(10), range(0, 10), and range(0, 10, 1) – They will produce the same results, however there is difference as follows:

for i in range(10):

Here, the variable i will iterate with values from 0 to 9 , as default starting value here is 0 and ending value is 9 (exclusive of last number). Default step value is 1, ie. Value of i is getting incremented by 1 during each iteration.

for i in range(0,10):

Here, we are supplying the starting value 0 explicitly, therefore, it will also iterate with values from 0 to 9. Again, the default step value is also 1.

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

Here, we are supplying the starting value 0 and also providing the increment step of 1, explicitly.

1. **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 for Loop:

for i in range(0,11,1):

print(i)

Using While loop:

i=0

while i<11:

print(i)

i=i+1

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

Ans:

We would create a python file (module) with the following code:

def bacon():

print(“This is print from function named bacon of spam module !!!")

We save the above file with name spam.py

After this, we create another file, say test\_spam.py and write the following code to call the bacon function after importing the spam module:

import spam

spam.bacon()