**FSDS MAY BATCH 2022(Python Basics -2)**

**Submitted by: Shubham Tiwari**

Q1.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** and **False.**

Q2. What are the three different types of Boolean operators?

Ans: The three different types of Boolean operators in Python are:

1. **‘and’** operator: returns True if both the operands are True
2. **‘or’** operator: returns True if either of the operands is True
3. **‘not’** operator: negates the truth value of the operand.

Q3. 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: Here are the truth tables for each Boolean operator in Python:

1. **‘and’** operator:

**A B A and B**

True True True

True False False

False True False

False False False

2**. ‘or’** operator:

**A B A or B**

True True True

True False True

False True True

False False False

**‘not’** operator:

**A not A**

True False

False True

Q4. 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) = True and False = False**

**not (5 > 4) = not True = False**

**(5 > 4) or (3 == 5) = True or False = True**

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

**(True and True) and (True == False) = True and False = False**

**(not False) or (not True) = True or False = True**

Q5. What are the six comparison operators?

Ans: The six comparison operators in Python are:

1. **==** (equal to)
2. **!=** (not equal to)
3. **<** (less than)
4. **>** (greater than)
5. **<=** (less than or equal to)
6. **>=** (greater than or equal to)

Q6. How do you tell the difference between the equal to and assignment operators? Describe a condition and when you would use one.

Ans: In Python, the equal to operator (==) is used for comparison and returns True or False based on whether the values being compared are equal. The assignment operator (=) is used for assignment, which assigns a value to a variable.

For example:

**x = 5**

**y = 3**

**print(x == y) # False**

We would use the equal to operator when you want to compare values, and the assignment operator when you want to assign a value to a variable.

Q7. 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:

1. **Initialization block: "spam = 0".**
2. **Conditional statement block: "if spam == 10" and "if spam > 5".**
3. **Print statements block: "print('eggs')", "print('bacon')", "print('ham')", "print('spam')", and "**print**('spam')".**

Q8. 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!")**

Q9.If your programme is stuck in an endless loop, what keys you’ll press?

Ans: The key combination **Ctrl + C** is used to break out of an infinite loop in a program.

Q10. How can you tell the difference between break and continue?

Ans: ‘**break**’statement is used to exit a loop prematurely when a certain condition is met, whereas ‘**continue’** statement is used to skip the current iteration of a loop and continue with the next iteration when a certain condition is met.

In other words, ‘**break’** terminates the loop and moves on to the next statement outside of the loop, whereas ‘**continue’** only skips the current iteration and continues with the next iteration of the loop.

Q11. 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(10)**, **range(0, 10)**, and **range(0, 10, 1)** generate equivalent sequences of integers, but with different arguments passed to the **range** function:

* **range(10)** generates a sequence of integers from 0 up to (but not including) 10: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**.
* **range(0, 10)** generates the same sequence as **range(10)**.
* **range(0, 10, 1)** generates the same sequence as **range(10)** and **range(0, 10)**. The third argument **1** specifies the step size to take in generating the sequence, and in this case, it is not necessary as the default step size is 1.

So in conclusion, all three produce the same sequence of integers and can be used interchangeably in a for loop.

Q12. 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: **for loop**

for i in range(1, 11):

print(i)

**while loop**

i = 1

while i <= 10:

print(i)

i += 1

Q13. If you had a function named bacon() inside a module named spam, how would you call it after importing spam?

Ans: To call the function **bacon()** inside the module **spam**, we would use the following syntax after importing the module

**import spam**

**spam.bacon()**