Python Assignment 2

1. The two values of Boolean type in python are ‘True’ and ‘False’. In python these values are written exactly as shown, with uppercase first letter and lowercase for the rest of the word.
2. Logical AND (represented as ‘and’) – This returns True if both operands are True

Eg: True and True 🡪 True

True and False 🡪 False

Logical OR (represented as ‘or’) – This returns True if any one of the operands is True

Eg: True or False 🡪 True

False or False 🡪 False

Logical NOT (represented as ‘not’) – It negates the value of the operand. If the operand is True result is False and vice-versa.

Eg: not True 🡪 False

not False 🡪 True

1. Logical AND (represented as ‘and’)

|  |  |  |
| --- | --- | --- |
| Operand 1 | Operand 2 | Result |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Logical OR (represented as ‘or’)

|  |  |  |
| --- | --- | --- |
| Operand 1 | Operand 2 | Result |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 2 |

Logical NOT (represented as ‘not’)

|  |  |
| --- | --- |
| Operand | Result |
| 0 | 1 |
| 1 | 0 |

1. (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 or False) 🡪 not(True) 🡪 False

(True and True) and (True==False) 🡪 True and False 🡪 False

(not False) or (not True) 🡪 True or False 🡪 True

1. The six comparison operators are:

* > (greater than) – Checks if left operand is greater than right
* < (less than) – Checks if left operand is less than right
* >= (greater than or equal to) – Checks if left operand is greater than or equal to right
* <= (less than or equal to) – Checks if left operand is less than or equal to right
* == (equal) – Checks if two values are equal
* != (not equal to) – Checks if two values are not equal

1. Equal to (==) – This is used to compare two values and check if they are equal. Returns True if values are equal and False otherwise

Assignment operator (=) – It is used to assign a value to a variable

Eg:

#Assignment operators

X = 10

Y = 5

#Equal to operator

print(X==Y) 🡪 Returns False as both are not equal

1. The three blocks in this code are:

Block 1:

if spam == 10:

print(‘eggs’)

Block 2:

if spam > 5:

print(‘bacon’)

Block 3:

else:

print(‘ham’)

print(‘spam’)

print(‘spam’)

1. Code:

spam = 1

if spam==1:

print(“Hello”)

elif spam ==2:

print(“Howdy”)

else:

print(“Greetings!”)

1. Ctrl + C 🡪 It causes KeyboardInterrupt exception.
2. The **break** statement is used to abruptly terminate the execution of a loop. When encountered within a loop, the break statement causes a loop exit immediately. After the break statement is encountered the program execution continues with the code that follows the loop.

**continue** statement is used to skip the remaining code within the current iteration of a loop and moves to next iteration. When the continue statement is encountered, it jumps to the next iteration without exceeding any further statements in the current iteration.

1. Everything mentioned here outputs the same sequence but is different ways of representing a range()

range(10) 🡪 By default starts with 0 and ends at 9 has default increment of 1

range(0,10) 🡪 This mentiones that the starting is at 0 and ends at 9 with default increment of 1

range(0,10,1) 🡪 This also mentiones step value as 1 but returns the same sequence from 0 to 9

1. for loop:

for i in range(1,11):

print(i)

while loop:

i = 1

while i<=10:

print(i)

i = i+1

1. We need to call the function as spam.bacon()