Python Basics Assignment 2

1.What are the two values of the boolean data types? how do you write them ?

Ans: Two values of the boolen data types are True and False. We have to use capital T and F and with the rest of the word in lowercase. The type() of both False and True is bool. The type bool is built in, meaning it's always available in Python and doesn't need to be imported. However, the name itself isn't a keyword in the language.

```
a=True
b=False
print(a,type(a))
print(b,type(b))
```

```
True <class 'bool'>
False <class 'bool'>
```

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

Ans: Boolean operators form the basis of mathematical sets and database logic. They connect our search words together to either narrow or broaden our set of results.The three differnt types of Boolean operators in python are: are, and, not Example: a>40 and b>40

```
a=1000
b=2000
print(a>500 and b>1000) # Example of boolean and
print(a>200 or b>1000) # Example of boolean or
print(not(a>11)) # Example of boolean not
```

```
True
True
False
```

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 evaluate)

Ans: The Truth tables for the boolean tables are as follows:

- **Truth Table for and operator**
  True and True --> True
  True and False --> False
  False and True --> False
  False and False --> False

- **Truth Table for or operator**
  True or True --> True
  True or False --> True

False or True --> True
False or False --> False

- **Truth Table for not operator**
  True not --> False False not --> 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:

```
print((5>4)and(3==5)) # False
print(not(5>4)) # False
print((5>4)or(3==5)) # True
print(not((5>4)or(3==5))) # False
print((True and True)and(True==False)) # False
print((not False)or(not True)) # True
```

```
False
False
True
False
False
True
```

5. What are the six comparison operators?

Ans: The Six Comparision Operators available in python are:
== , != , < , > , <= , =>

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

Ans: == is the equal to operator that compares two values and evaluates to a Boolean, while = is that assignment operator that stores a value in a variable.

```
b=10 # Assigning operator that stores 3 value in a variable a
if b==10:#comparing values of a varible value and 3
   print(b==3)
```

```
False
```

7. 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: In Python, code block refers to a collection of code that is in the same block or indent. This is most commonly found in classes, functions, and loops. Answer for the question is:

ham

spam

spam

```
spam = 0
if spam == 10:
    print('eggs')  # block #1
if spam > 5:
    print('bacon')  # block #2
else:
    print('ham')  # block #3
print('spam')
print('spam')
```

```
ham
spam
spam
```

8. 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:

```
def spamCode(spam):
    if spam==1:
        print('Hello')
    elif spam==2:
        print('Howdy')
    else:
        print('Greetings')

spamCode(2)
spamCode(3)
spamCode(1)
```

Howdy
Greetings
Hello

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

Ans: To stop a program stuck in an infinite loop, we press Ctrl-c.

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

Ans: The break statement will move the execution outside the loop if break condtion is satisfied whereas the continue statement will move the execution to the start of the loop. Example of break and continue is mentioned below:

```
# Use of break statement inside the loop

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```

```
s
t
r
The end
```

```
# Program to show the use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

```
s
t
r
n
g
The end
```

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

Ans: The Differences are as follows:

1. The ***range(10)*** call range from 0 to 9 (but not include 10)
2. The ***range (0,10)*** explicitly tells the loop to start at 0
3. The ***range(0,10,1)*** explicitly tells the loop to increase the variable by 1 on each iteration

```
# printing a number
for i in range(10):
    print(i, end=" ")
print()
```

0 1 2 3 4 5 6 7 8 9

```
# performing sum of numbers
sum = 0
for i in range(0, 10):
    sum = sum + i
print("Sum of numbers :", sum)
```

Sum of numbers : 45

```
# performing sum of numbers
sum = 0
for i in range(0, 10, 1):
    sum = sum + i
print("Sum of numbers :", sum)
```

Sum of numbers : 45

12. 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:

```
print('-'*10,'Using For Loop','-'*10)
for i in range(1,11):
    print(i, end=" ")
print('\n')
print('-'*10,'Using While Loop','-'*10)
i=1
while i<=10:
    print(i, end=" ")
    i+=1
```

---------- Using For Loop ----------
1 2 3 4 5 6 7 8 9 10

---------- Using While Loop ----------
1 2 3 4 5 6 7 8 9 10

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

Ans: This function can be called with spam.bacon()