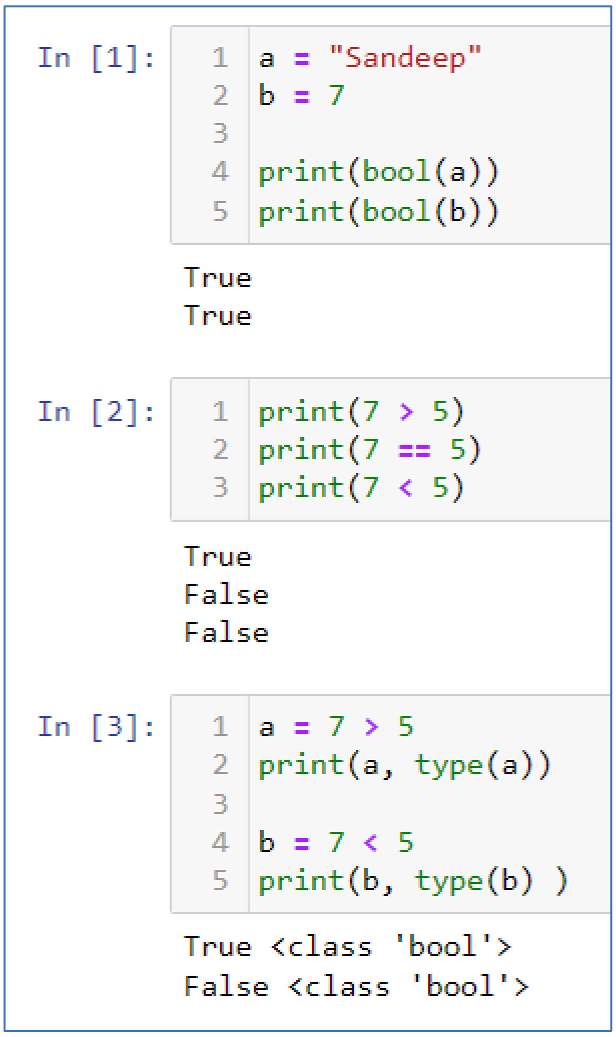
1.What are the two values of the Boolean data type? How do you write them?

Two values of Boolean data type are **True** or **False**.

They can be written as,..



```
In [1]:  a = "Sandeep"
         b = 7

         print(bool(a))
         print(bool(b))

         True
         True

In [2]:  print(7 > 5)
         print(7 == 5)
         print(7 < 5)

         True
         False
         False

In [3]:  a = 7 > 5
         print(a, type(a))

         b = 7 < 5
         print(b, type(b) )

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

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

The three different types of Boolean operators are AND, OR, NOT.

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).

Truth Table for Boolean Operator, AND

A	B	A and B
TRUE ✓	TRUE ✓	TRUE ✓
TRUE ✓	FALSE ×	FALSE ×
FALSE ×	TRUE ✓	FALSE ×
FALSE ×	FALSE ×	FALSE ×

Truth Table for Boolean Operator, OR

A	B	A or B
TRUE ✓	TRUE ✓	TRUE ✓
TRUE ✓	FALSE ×	TRUE ✓
FALSE ×	TRUE ✓	TRUE ✓
FALSE ×	FALSE ×	FALSE ×

Truth Table for Boolean Operator, NOT

A	not A
FALSE ×	TRUE ✓
TRUE ✓	FALSE ×

4. What are the values of the following expressions?

(5 > 4) and (3 == 5) **FALSE**

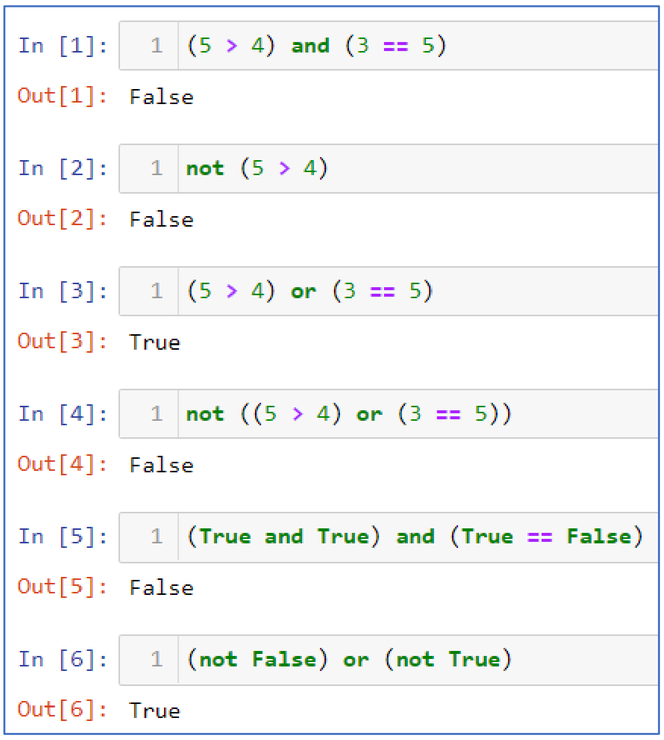
not (5 > 4) **FALSE**

(5 > 4) or (3 == 5) **TRUE**

not ((5 > 4) or (3 == 5)) **FALSE**

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

(not False) or (not True) **TRUE**



```
In [1]:  (5 > 4) and (3 == 5)
Out[1]:  False

In [2]:  not (5 > 4)
Out[2]:  False

In [3]:  (5 > 4) or (3 == 5)
Out[3]:  True

In [4]:  not ((5 > 4) or (3 == 5))
Out[4]:  False

In [5]:  (True and True) and (True == False)
Out[5]:  False

In [6]:  (not False) or (not True)
Out[6]:  True
```

5. What are the six comparison operators?

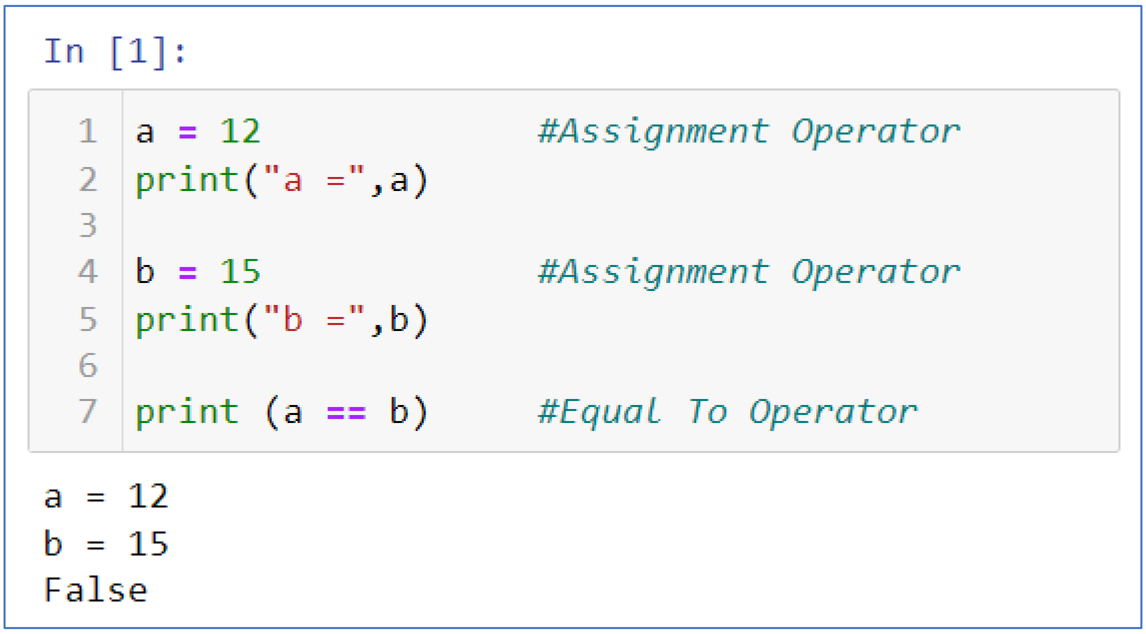
Python has six comparison operators, which are as follows:

- Less than (<)
- Less than or equal to (<=)
- Greater than (>)
- Greater than or equal to (>=)
- Equal to (==)
- Not equal to (!=)

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

If **'='** ('equal to' typed **once**) is used, then it is **"assignment"** operator.

If **'=='** ('equal to' typed **twice**) is used, then it is **"equal to"** operator.



```
In [1]:
a = 12              #Assignment Operator
print("a =",a)

b = 15              #Assignment Operator
print("b =",b)

print (a == b)      #Equal To Operator
```

```
a = 12
b = 15
False
```

7. Identify the three blocks in this code:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

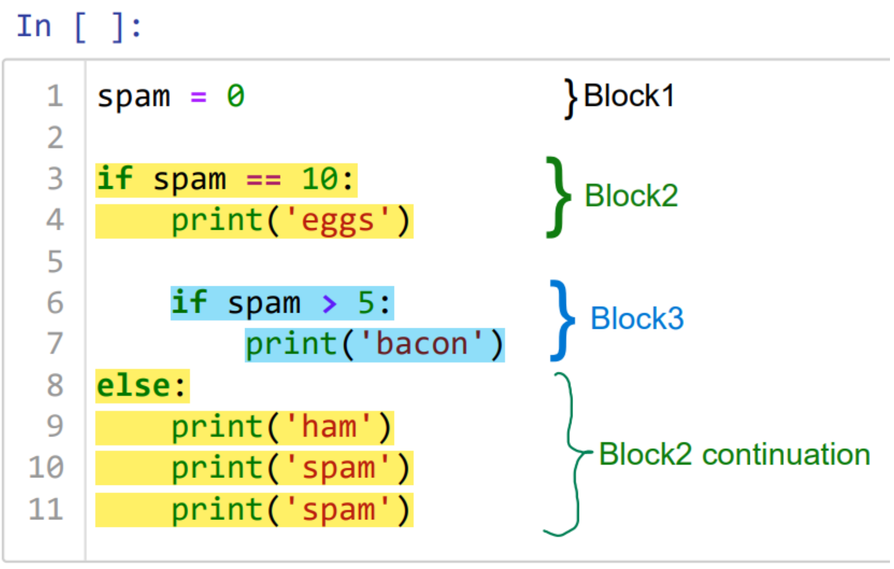
print('bacon')

else:

print('ham')

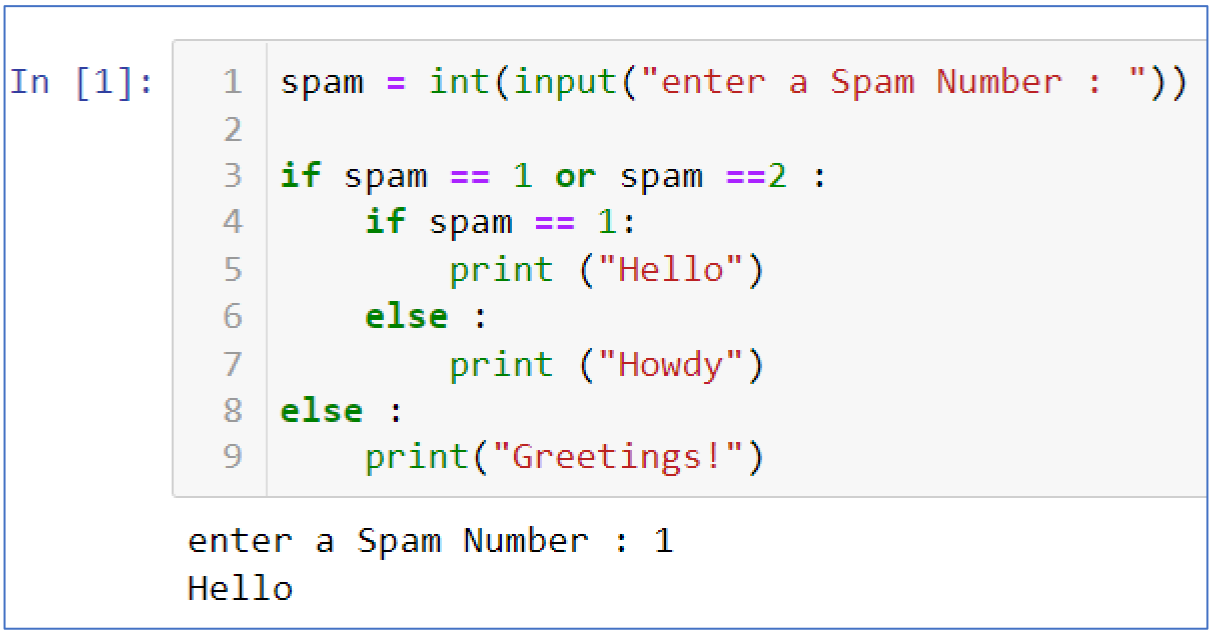
print('spam')

print('spam')



```
In [ ]:
spam = 0

if spam == 10:
    print('eggs')

    if spam > 5:
        print('bacon')
else:
    print('ham')
    print('spam')
    print('spam')
```

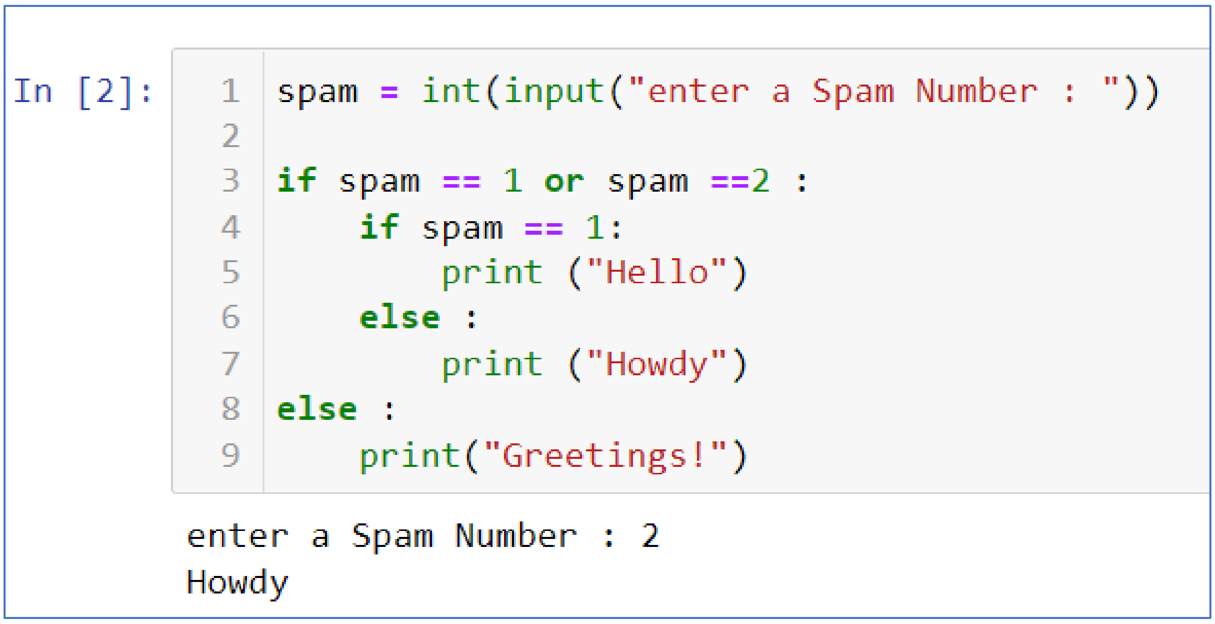
8. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! if anything else is stored in spam.



In [1]:

```python
spam = int(input("enter a Spam Number : "))

if spam == 1 or spam ==2 :
    if spam == 1:
        print ("Hello")
    else :
        print ("Howdy")
else :
    print("Greetings!")
```

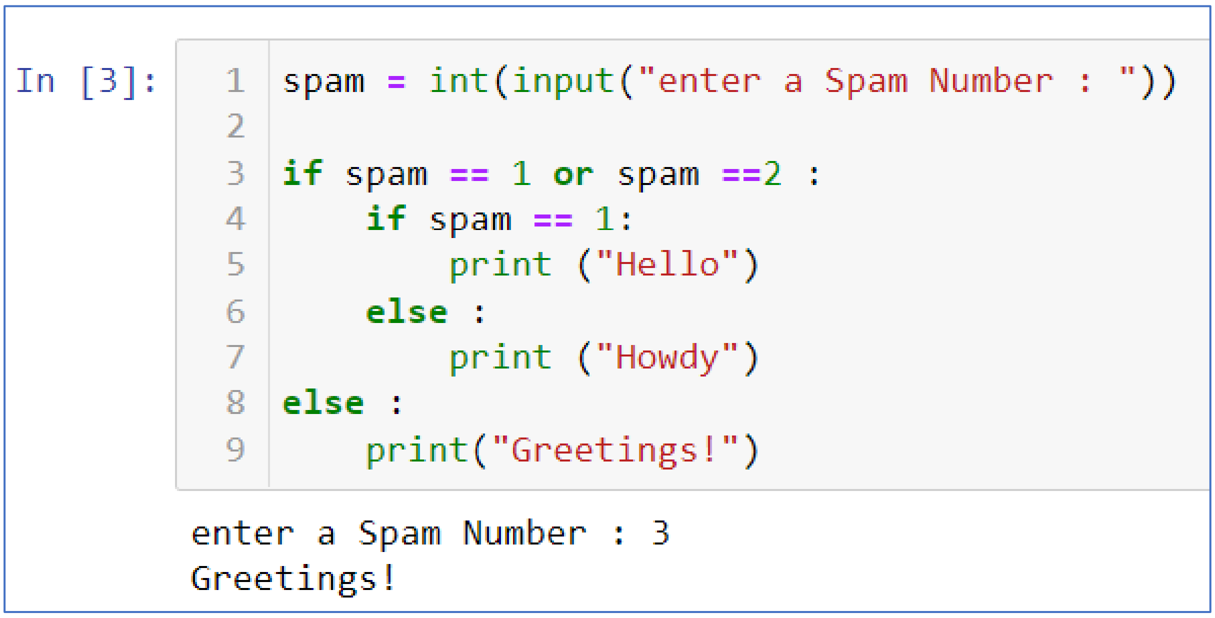
```
enter a Spam Number : 1
Hello
```



In [2]:

```python
spam = int(input("enter a Spam Number : "))

if spam == 1 or spam ==2 :
    if spam == 1:
        print ("Hello")
    else :
        print ("Howdy")
else :
    print("Greetings!")
```

```
enter a Spam Number : 2
Howdy
```



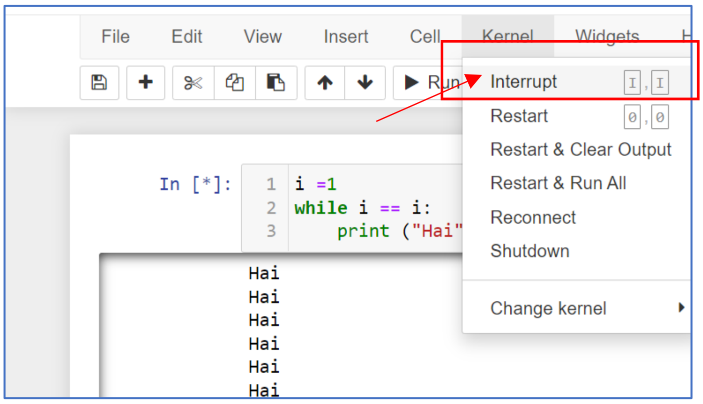
In [3]:

```python
spam = int(input("enter a Spam Number : "))

if spam == 1 or spam ==2 :
    if spam == 1:
        print ("Hello")
    else :
        print ("Howdy")
else :
    print("Greetings!")
```

```
enter a Spam Number : 3
Greetings!
```

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

There are two ways to come out of an endless loop. One of them is, by using keyboard, click 'I' letter key two times. Other way is, by using Mouse, in Jupyter notebook, in the toggle bar click Kernel, then click Interrupt.



```
In [*]:
i =1
while i == i:
    print ("Hai"
```

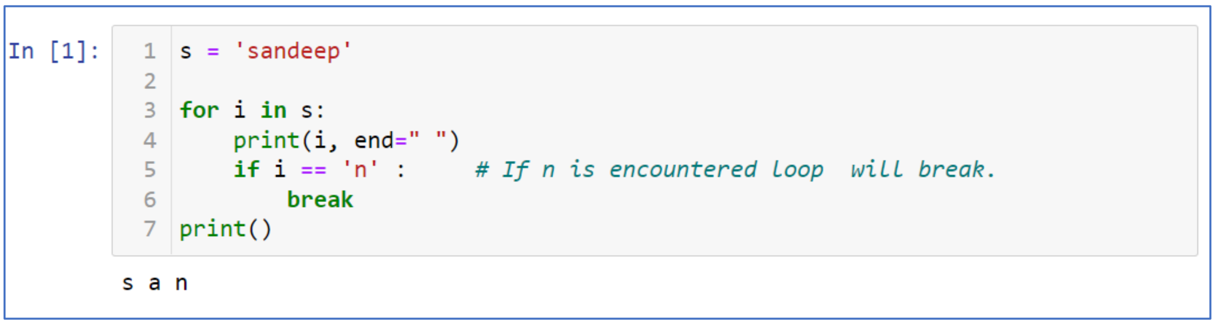
```
Hai
Hai
Hai
Hai
Hai
Hai
```

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

Break refers to **STOPPING** of **LOOP.**

Continue refers to **SKIPPING** of present **ITERATION.**

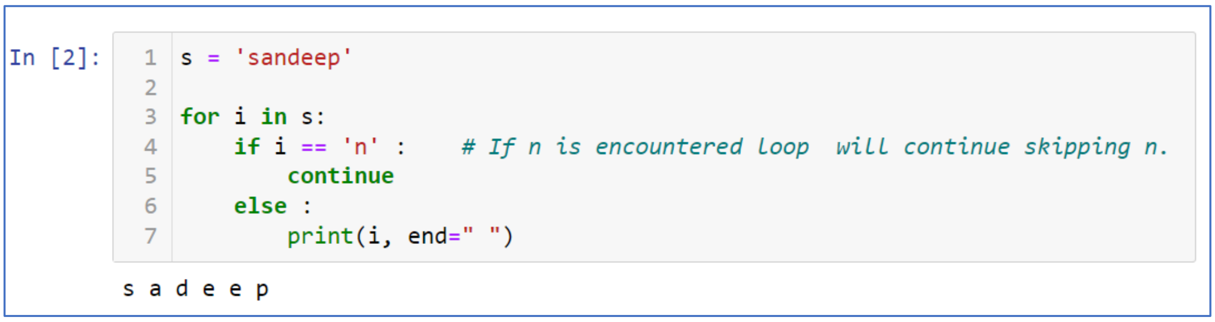
The **break** statement is used to terminate the loop or statement in which it is present. It **STOP**s the loop.



```
In [1]:
s = 'sandeep'

for i in s:
    print(i, end=" ")
    if i == 'n' :     # If n is encountered loop  will break.
        break
print()
```

```
s a n
```

continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop. It **SKIP**s the present iteration in the loop.



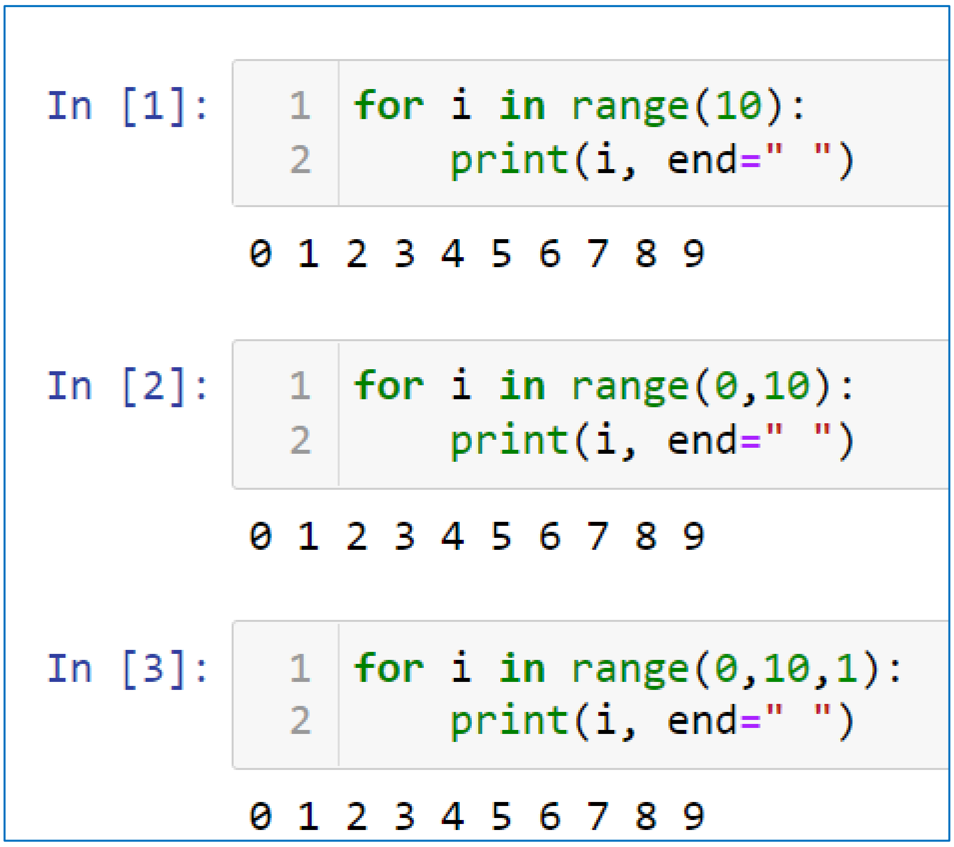
```
In [2]:
s = 'sandeep'

for i in s:
    if i == 'n' :    # If n is encountered loop  will continue skipping n.
        continue
    else :
        print(i, end=" ")
```

```
s a d e e p
```

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

The output is same for all three cases. But there is significant difference between all three of them on how they executes.

	Starting point	**Ending point PLUS one**	**Step value**
range(10)	**Not mentioned explicitly. But it is zero by default.**	**Only value mentioned in the brackets refers to End Point Plus One.**	**Not mentioned. But it is one by default.**
range(0,10)	**Mentioned explicitly. It will be the value BEFORE first comma.**	**Mentioned explicitly. It will be the value AFTER first comma.**	**Not mentioned. But it is one by default.**
range(0,10,1)	**Mentioned explicitly. It will be the value BEFORE first comma.**	**Mentioned explicitly. It will be the value AFTER first comma.**	**Mentioned explicitly. It will be the value AFTER second comma.**



```
In [1]:
for i in range(10):
    print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

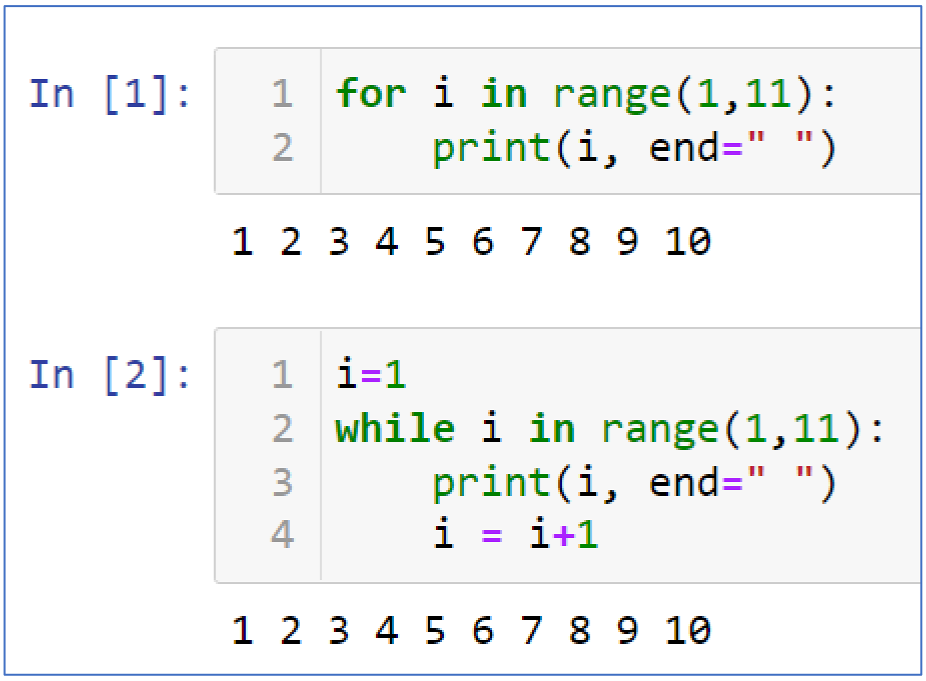
```
In [2]:
for i in range(0,10):
    print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

```
In [3]:
for i in range(0,10,1):
    print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

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.



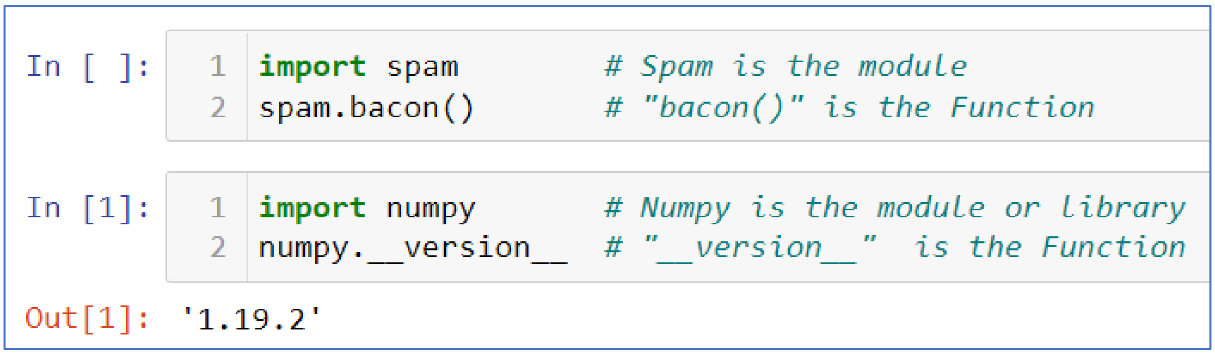
```
In [1]:
for i in range(1,11):
    print(i, end=" ")
```

```
1 2 3 4 5 6 7 8 9 10
```

```
In [2]:
i=1
while i in range(1,11):
    print(i, end=" ")
    i = i+1
```

```
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?



```
In [ ]:
import spam          # Spam is the module
spam.bacon()         # "bacon()" is the Function
```

```
In [1]:
import numpy         # Numpy is the module or library
numpy.__version__    # "__version__"  is the Function
```

```
Out[1]: '1.19.2'
```