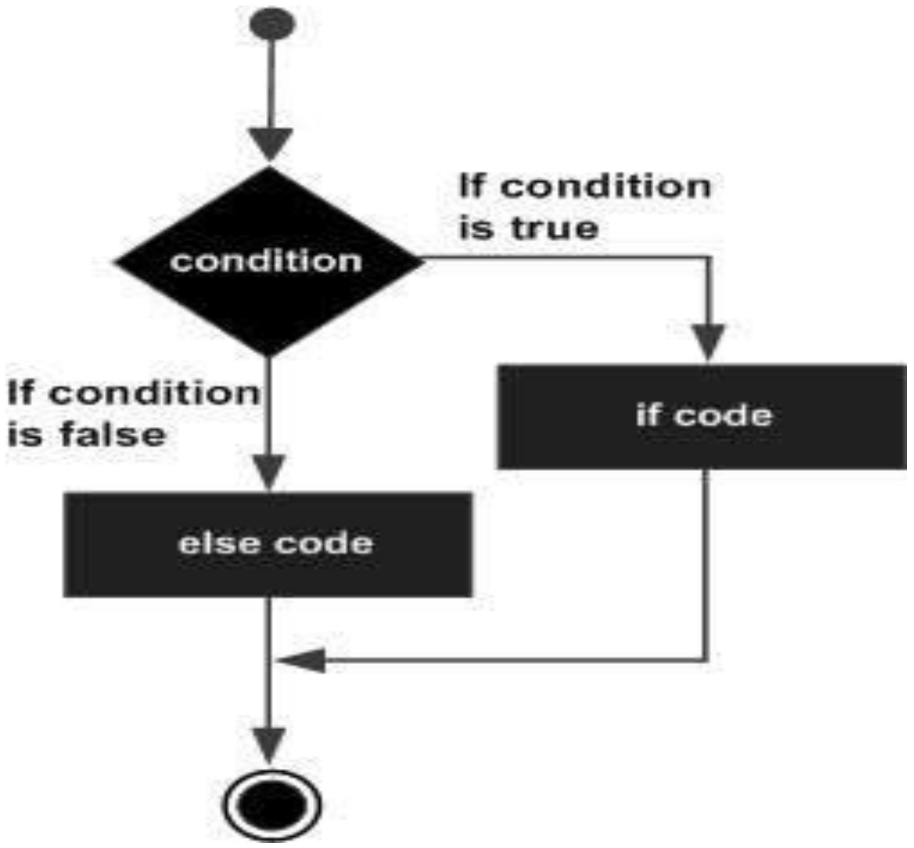
Conditional and Decision Making

If, else if, else Statement

Definition



- if Statements in Python allows us to tell the computer to perform alternative actions based on a certain set of results.
- Verbally, we can imagine we are telling the computer:
- "Hey if this case happens, perform some action"
- We can then expand the idea further with elif and else statements, which allow us to tell the computer:
- "Hey if this case happens, perform some action. Else if another case happens, perform some other action. Else-- none of the above cases happened, perform this action"

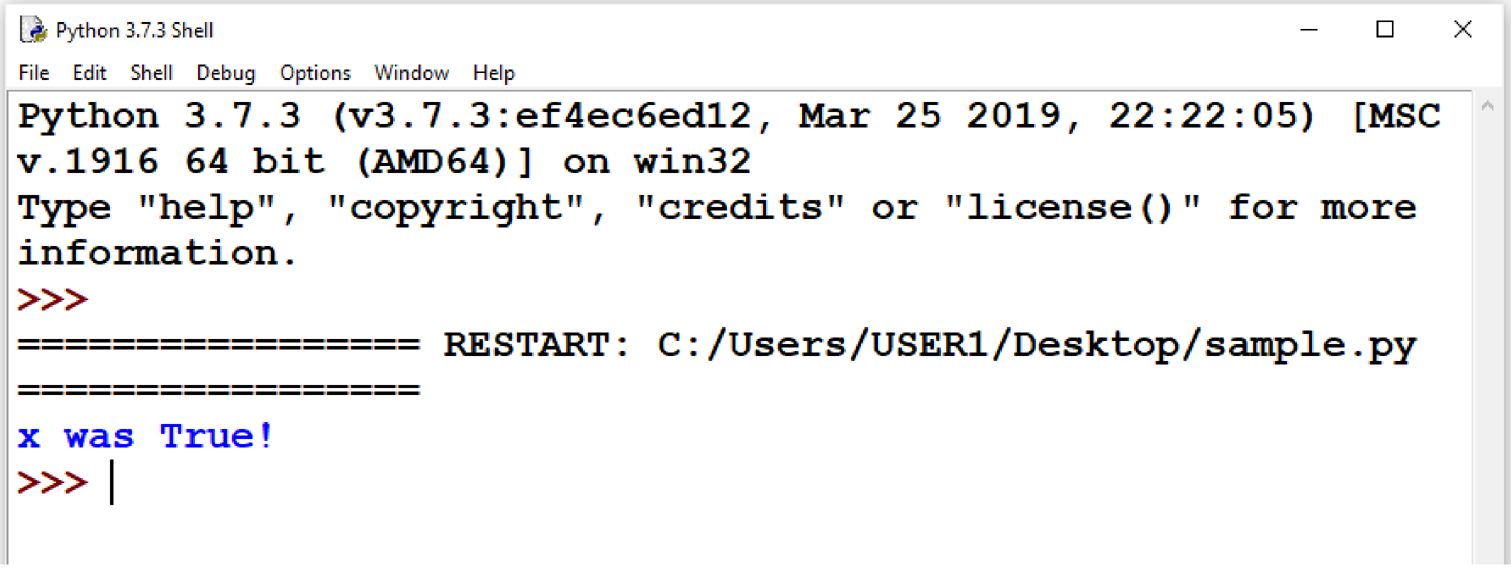
Syntax

```
If case1:
    perform action1
elif case2:
   action2
else:
    perform action 3
```

Example: 1

If the statement of x is true the block gets executed else it will go to the next block

```
x = False

if x==False:
    print ('x was True!')
else:
    print ('I will be printed in any case where x is not true')
```



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC
v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
================= RESTART: C:/Users/USER1/Desktop/sample.py
=================
x was True!
>>> |
```

If…elif..else

- Let's get a fuller picture of how far if, elif, and else can take us!
- We write this out in a nested structure. note for how the if,elif,and else line up in the code. This can help you see what if is related to what elif or else statements.

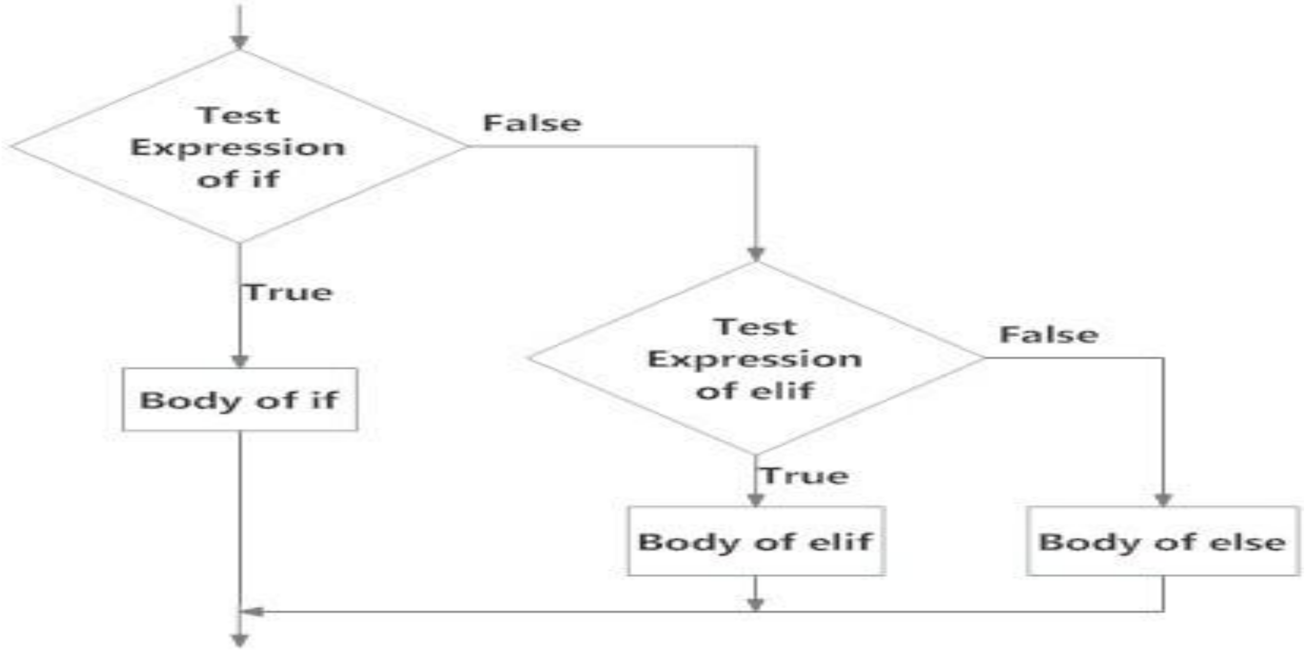
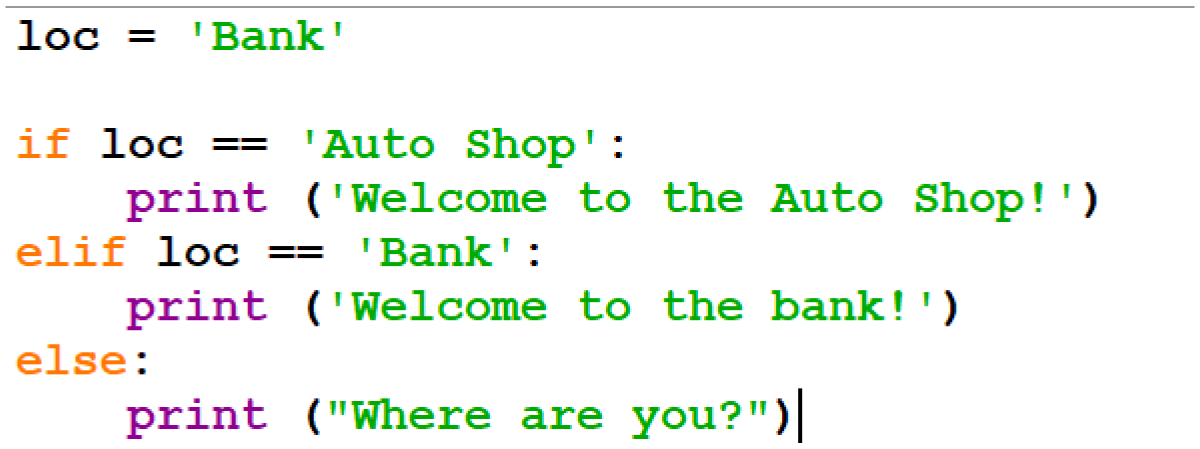


Fig: Operation of if...elif...else statement

Example 2:

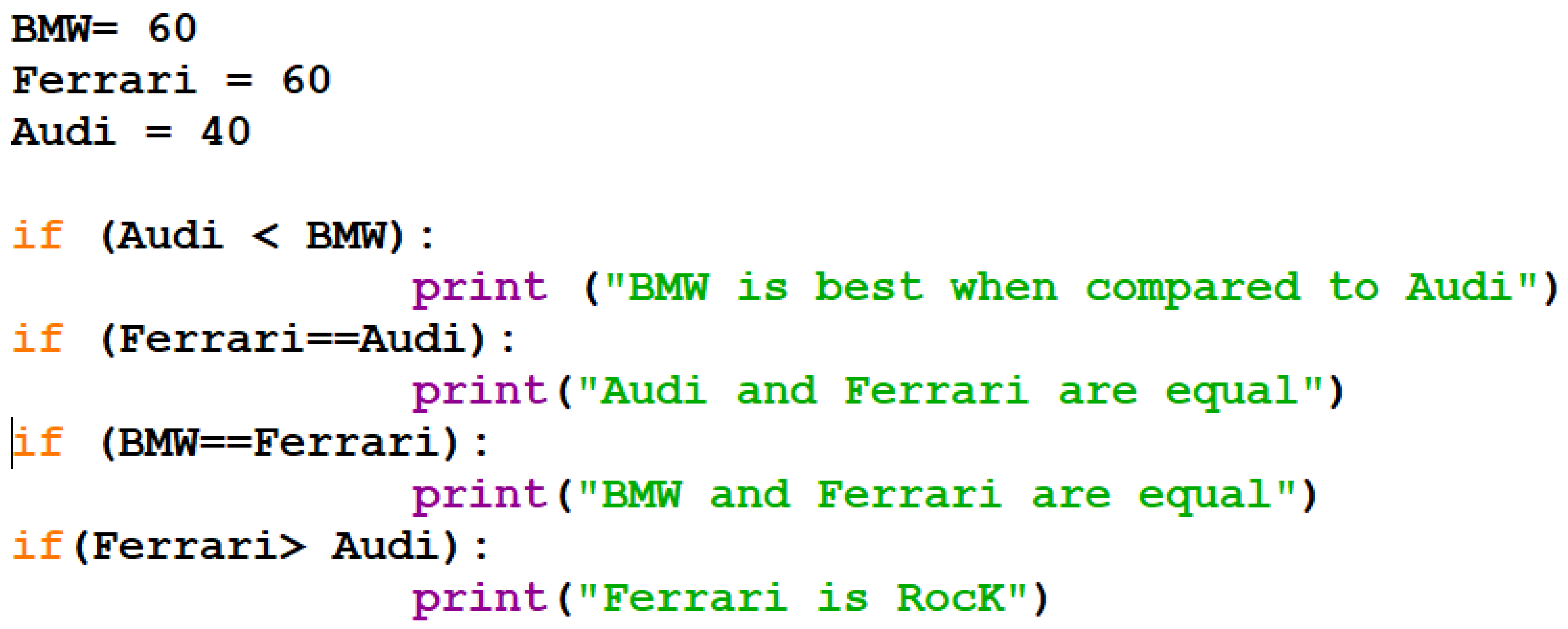
- So the following example execute based on the condition which is true, if one condition is true we will get the block



```python
loc = 'Bank'

if loc == 'Auto Shop':
    print ('Welcome to the Auto Shop!')
elif loc == 'Bank':
    print ('Welcome to the bank!')
else:
    print ("Where are you?")
```

only if:

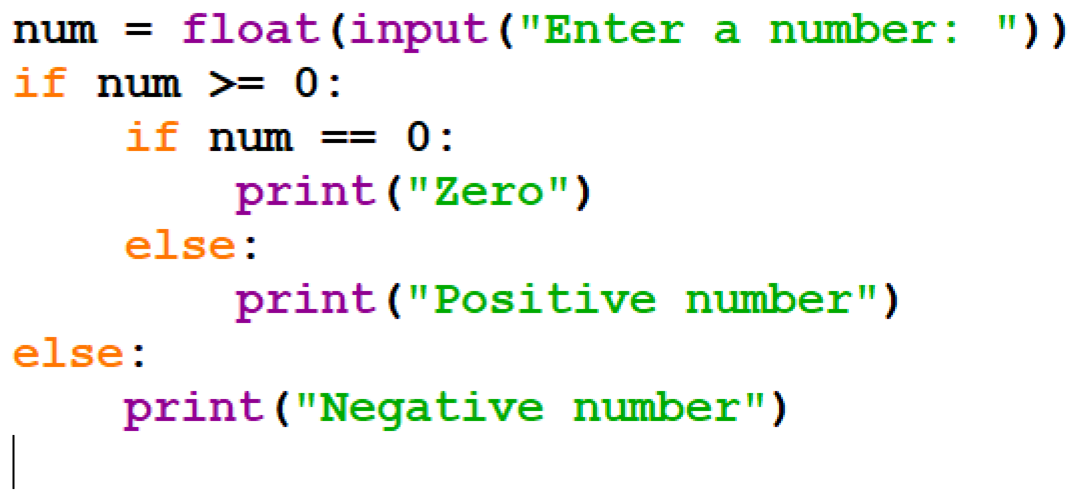
Using only if statement is possible but not advisable



```
BMW= 60
Ferrari = 60
Audi = 40

if (Audi < BMW):
               print ("BMW is best when compared to Audi")
if (Ferrari==Audi):
               print("Audi and Ferrari are equal")
if (BMW==Ferrari):
               print("BMW and Ferrari are equal")
if(Ferrari> Audi):
               print("Ferrari is RocK")
```

Nested if

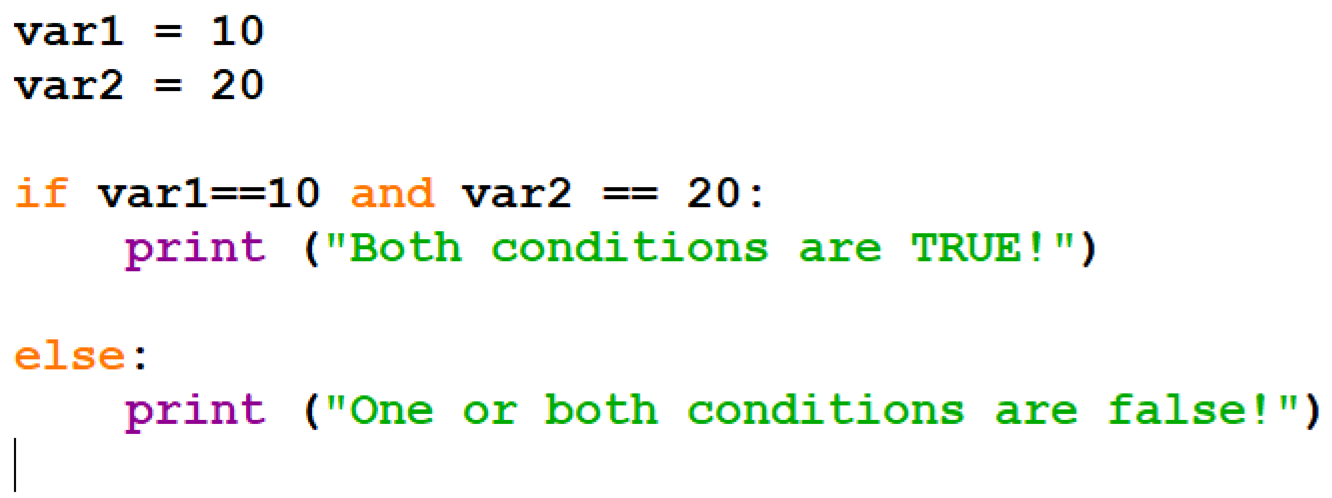
- Nested statement if the condition true inside the block we have multiple conditions to check



```python
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

and operator

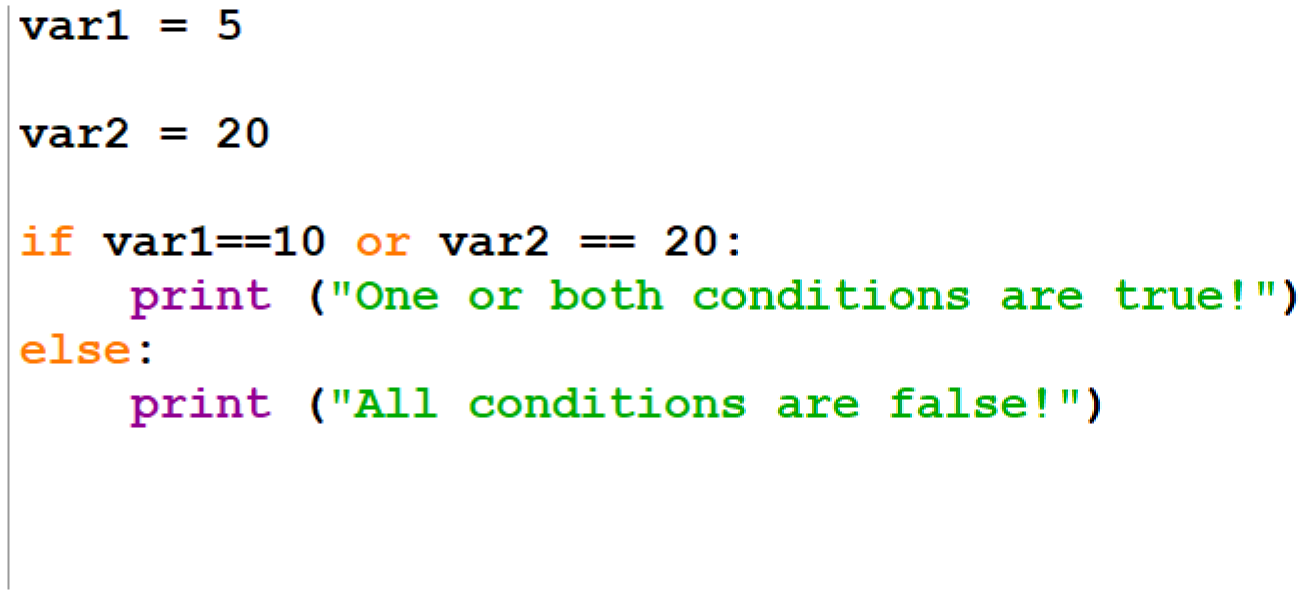
- if condition, the **“and” operator** is used.
- using the *‘and’* operator, all the conditions have to be true in order to execute the statements inside the if statement. If any of the conditions is false, the else part (if provided) will execute



```
var1 = 10
var2 = 20

if var1==10 and var2 == 20:
    print ("Both conditions are TRUE!")

else:
    print ("One or both conditions are false!")
```

Or Operator

- **‘or’ operator** is used in Python if else statement. Two conditions are given, so, if any of the conditions is true, the if part will execute:



```
var1 = 5

var2 = 20

if var1==10 or var2 == 20:
    print ("One or both conditions are true!")
else:
    print ("All conditions are false!")
```

Decision Making

For, while, break and continue

Definition

a **for-loop** (or simply **for loop**) is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

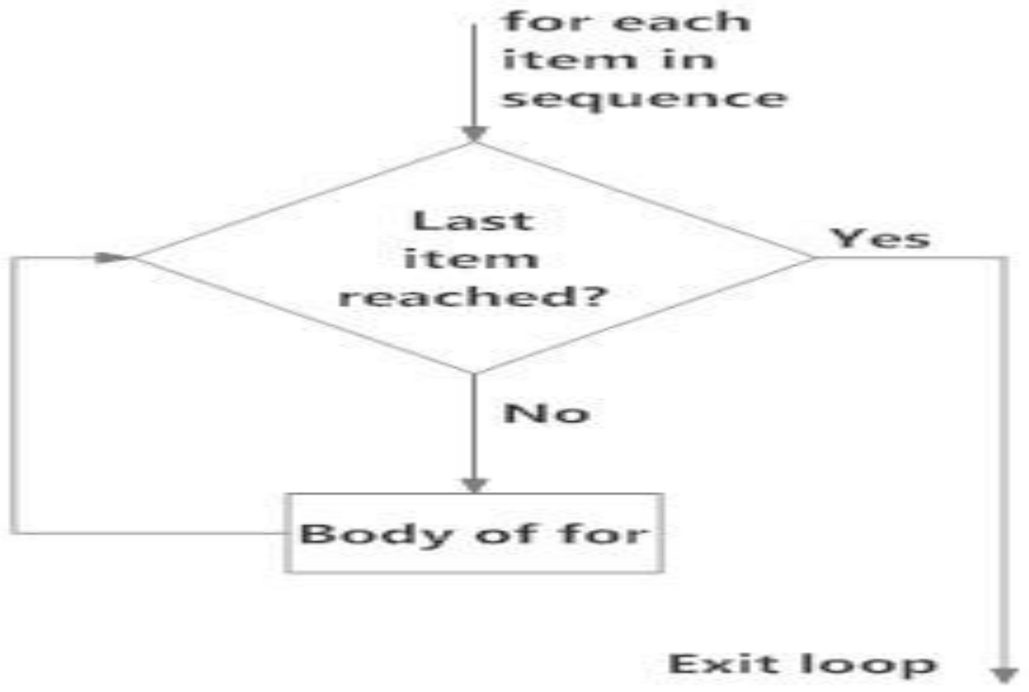


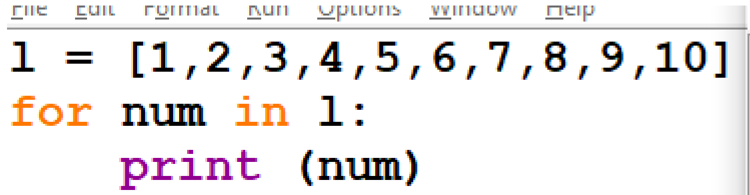
Fig: operation of for loop

Syntax

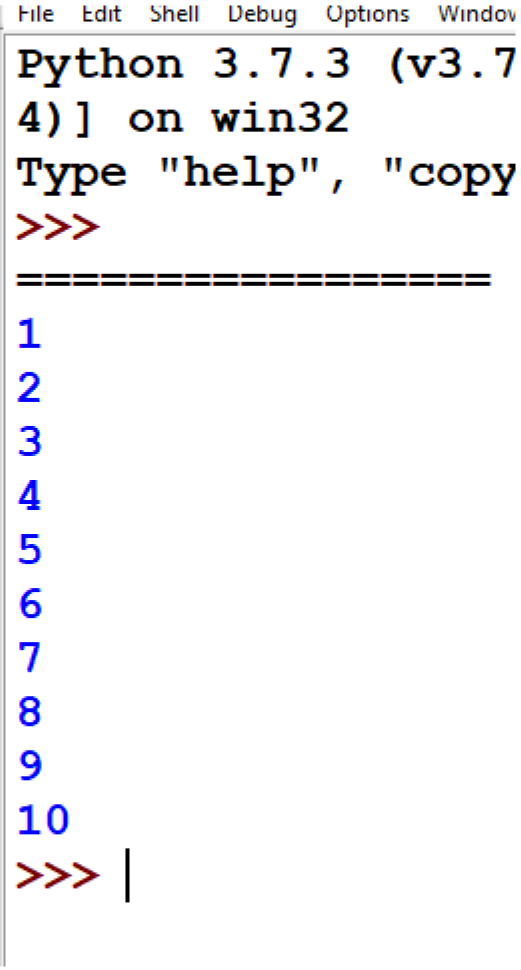
```
for val in sequence:
        Body of Statements
```

Example 1:

With List Data type until reach the last sequence

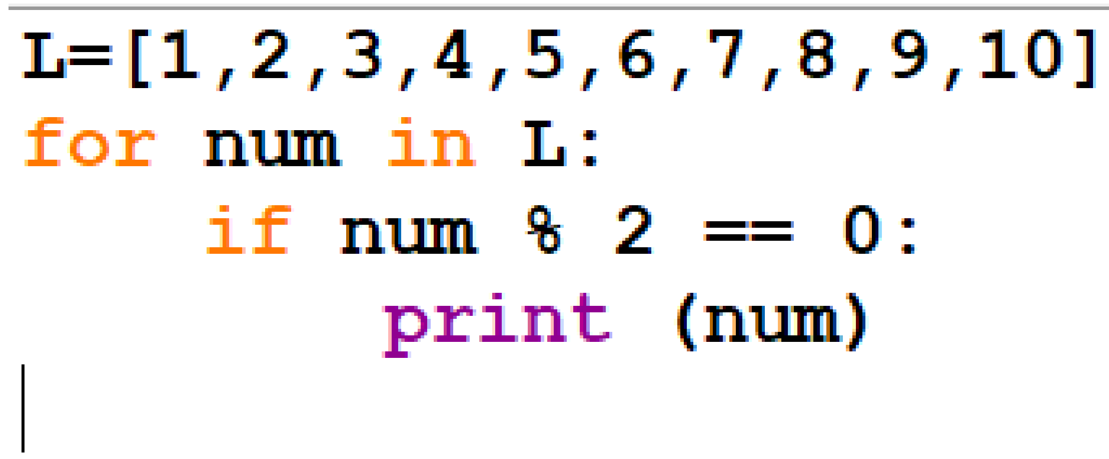


```
l = [1,2,3,4,5,6,7,8,9,10]
for num in l:
    print (num)
```



```
Python 3.7.3 (v3.7
4)] on win32
Type "help", "copy
>>>
===================
1
2
3
4
5
6
7
8
9
10
>>> |
```

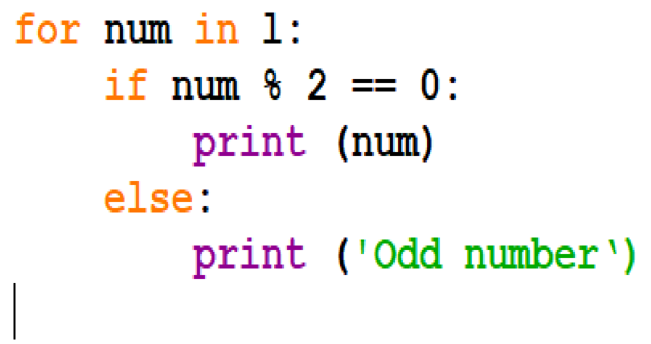
Print only even number in List



```
L=[1,2,3,4,5,6,7,8,9,10]
for num in L:
    if num % 2 == 0:
        print (num)
```

Print even numbers else print as odd number

- It will print the even number as a number and if it is odd number it print “odd number “

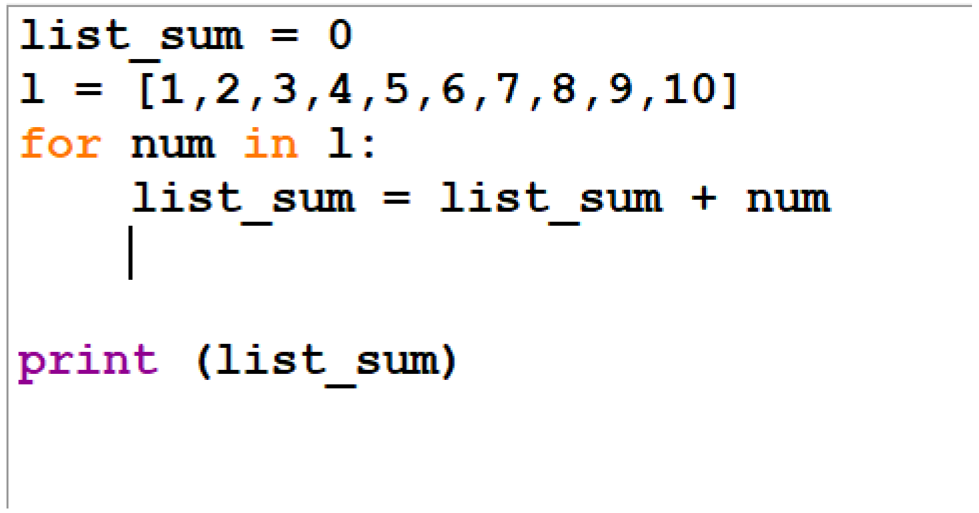


```
for num in l:
    if num % 2 == 0:
        print (num)
    else:
        print ('Odd number')
```

Summing up

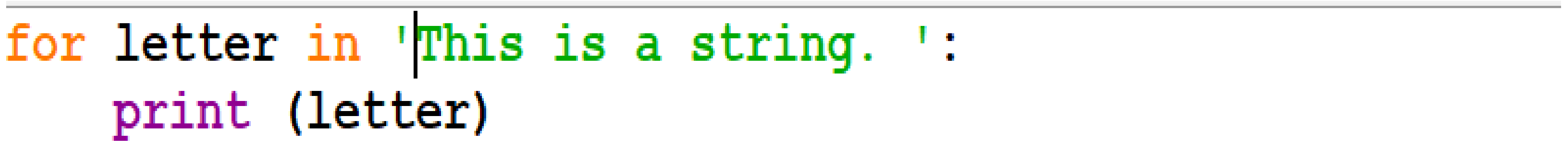
This example to understand the indent block

Try the print(list_sum) with ans without Indent



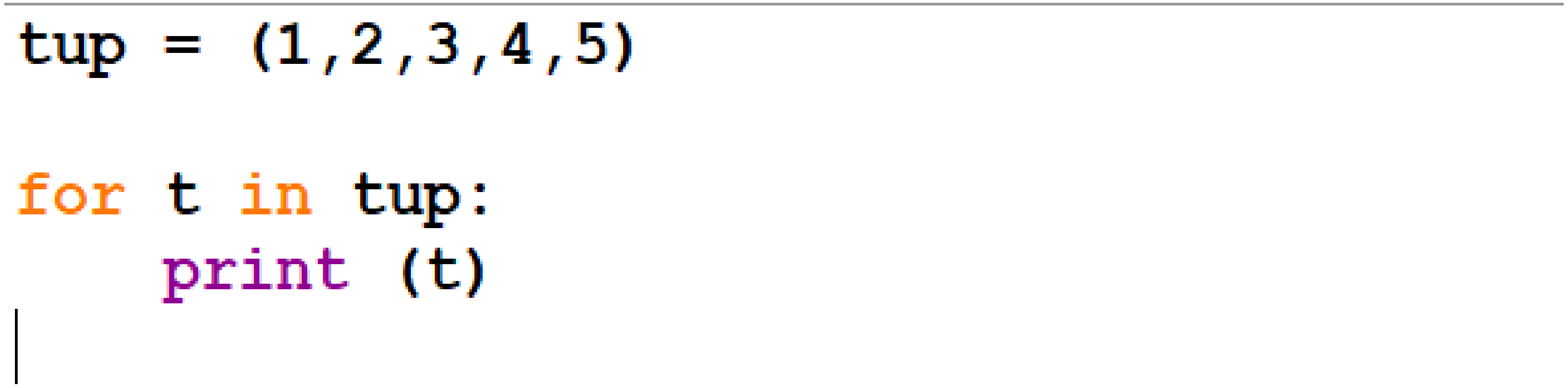
```
list_sum = 0
l = [1,2,3,4,5,6,7,8,9,10]
for num in l:
    list_sum = list_sum + num


print (list_sum)
```

For with String



```python
for letter in 'This is a string. ':
    print (letter)
```

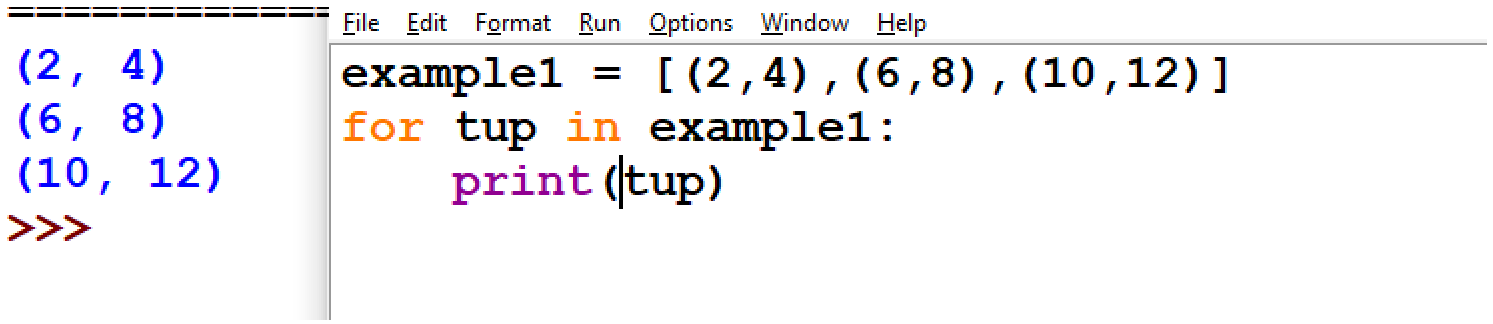
For with Tuple



```
tup = (1,2,3,4,5)

for t in tup:
    print (t)
```

Unpacking with For Loop

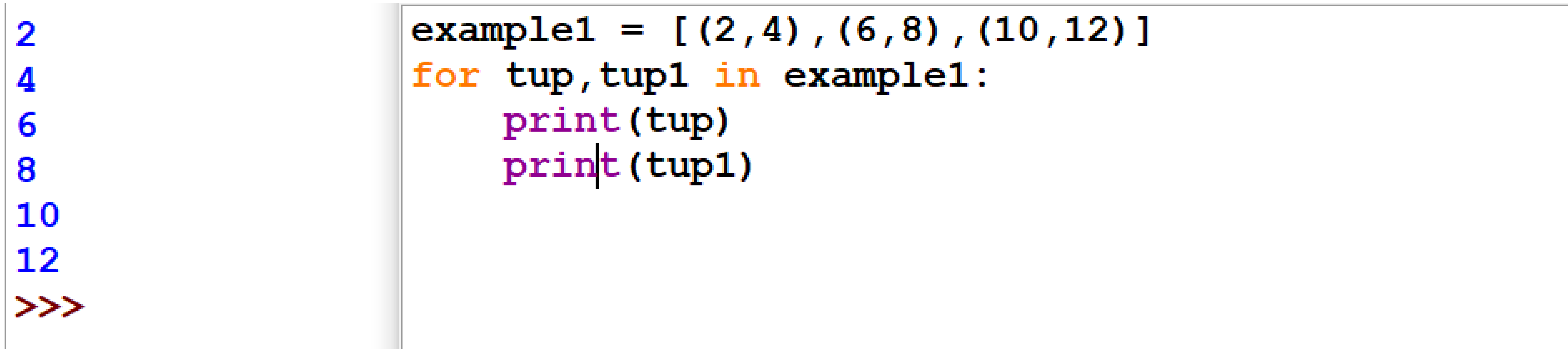
- This method is unpacking we have a list inside multiple tuples we can unpack with the help of for loop
- Example 1



```
============
(2, 4)
(6, 8)
(10, 12)
>>>
```

```python
example1 = [(2,4),(6,8),(10,12)]
for tup in example1:
    print(tup)
```

- Example 2

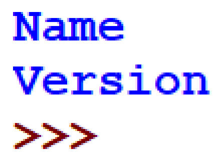


```
2
4
6
8
10
12
>>>
```

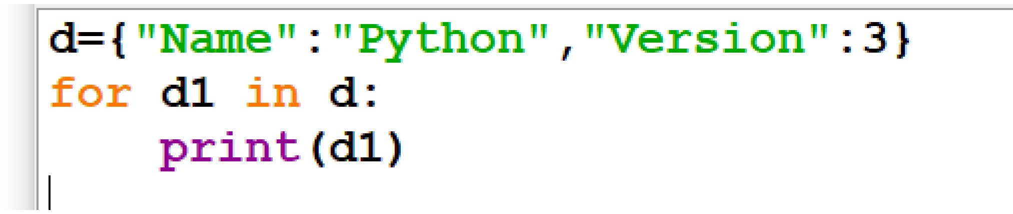
```python
example1 = [(2,4),(6,8),(10,12)]
for tup,tup1 in example1:
    print(tup)
    print(tup1)
```

For loop with Dictionary

- By default it gets the key name

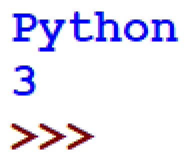


```
Name
Version
>>>
```

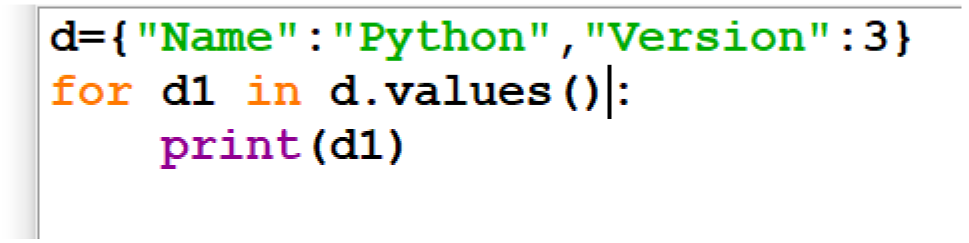


```
d={"Name":"Python","Version":3}
for d1 in d:
    print(d1)
```

- To get the values from dictionary

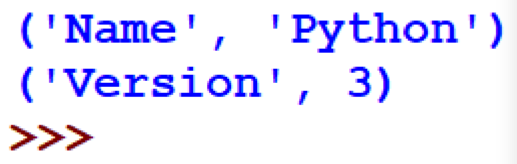


```
Python
3
>>>
```

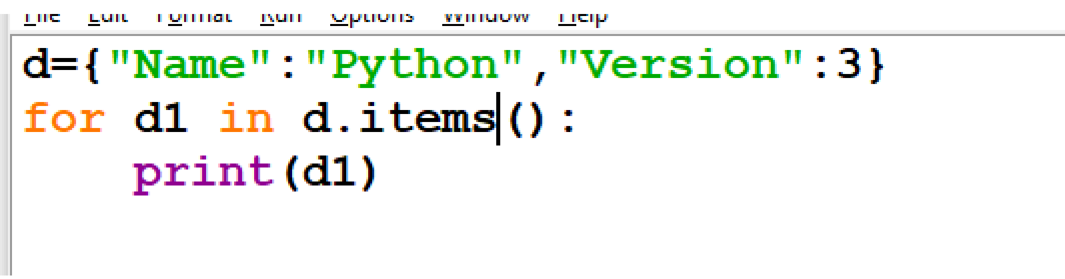


```
d={"Name":"Python","Version":3}
for d1 in d.values():
    print(d1)
```

- To get values and Keys



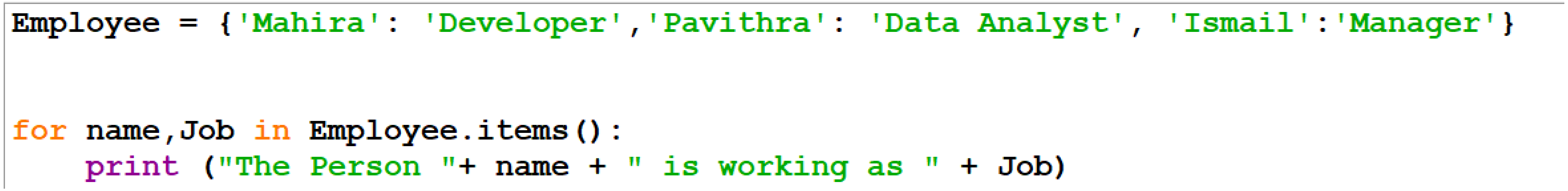
```
('Name', 'Python')
('Version', 3)
>>>
```



```
d={"Name":"Python","Version":3}
for d1 in d.items():
    print(d1)
```

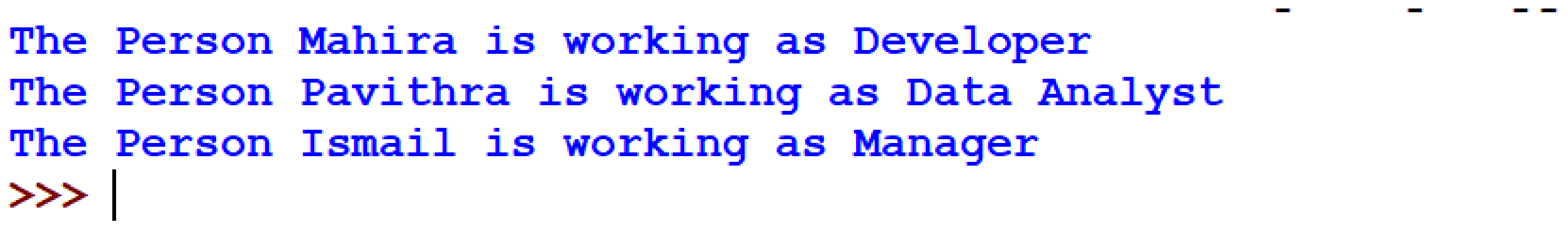
Dictionary with For Loop

- The Code



```
Employee = {'Mahira': 'Developer','Pavithra': 'Data Analyst', 'Ismail':'Manager'}


for name,Job in Employee.items():
    print ("The Person "+ name + " is working as " + Job)
```

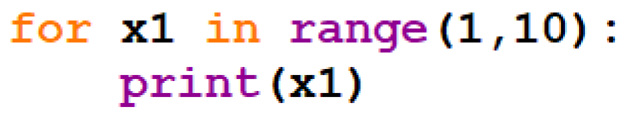
- The Output



```
The Person Mahira is working as Developer
The Person Pavithra is working as Data Analyst
The Person Ismail is working as Manager
>>> 
```

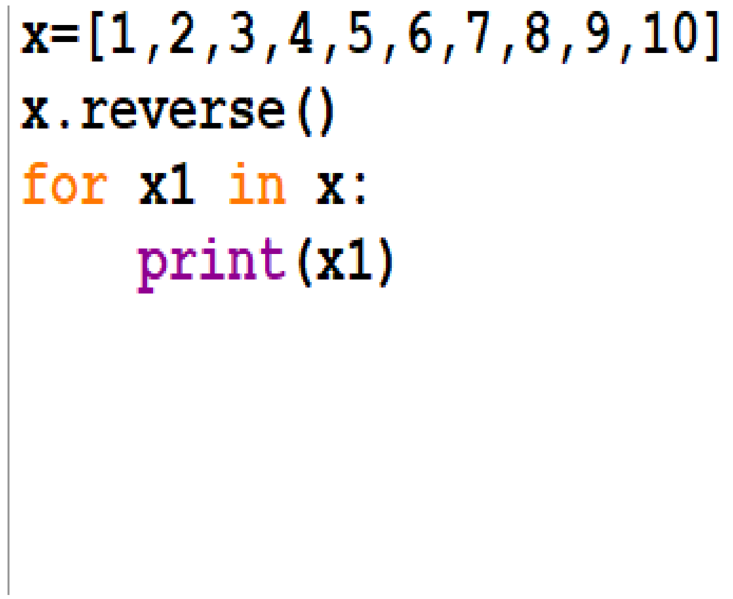
Example 11

- Range is an in build function to print a numbers in a particular range



```
for x1 in range(1,10):
    print(x1)
```

Print list in reverse order



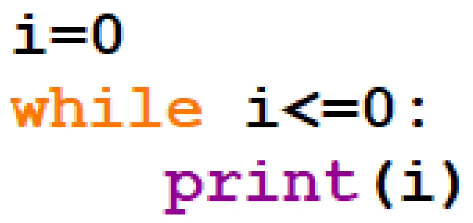
```python
x=[1,2,3,4,5,6,7,8,9,10]
x.reverse()
for x1 in x:
    print(x1)
```

While Loop

It will keep on executing until it becomes false

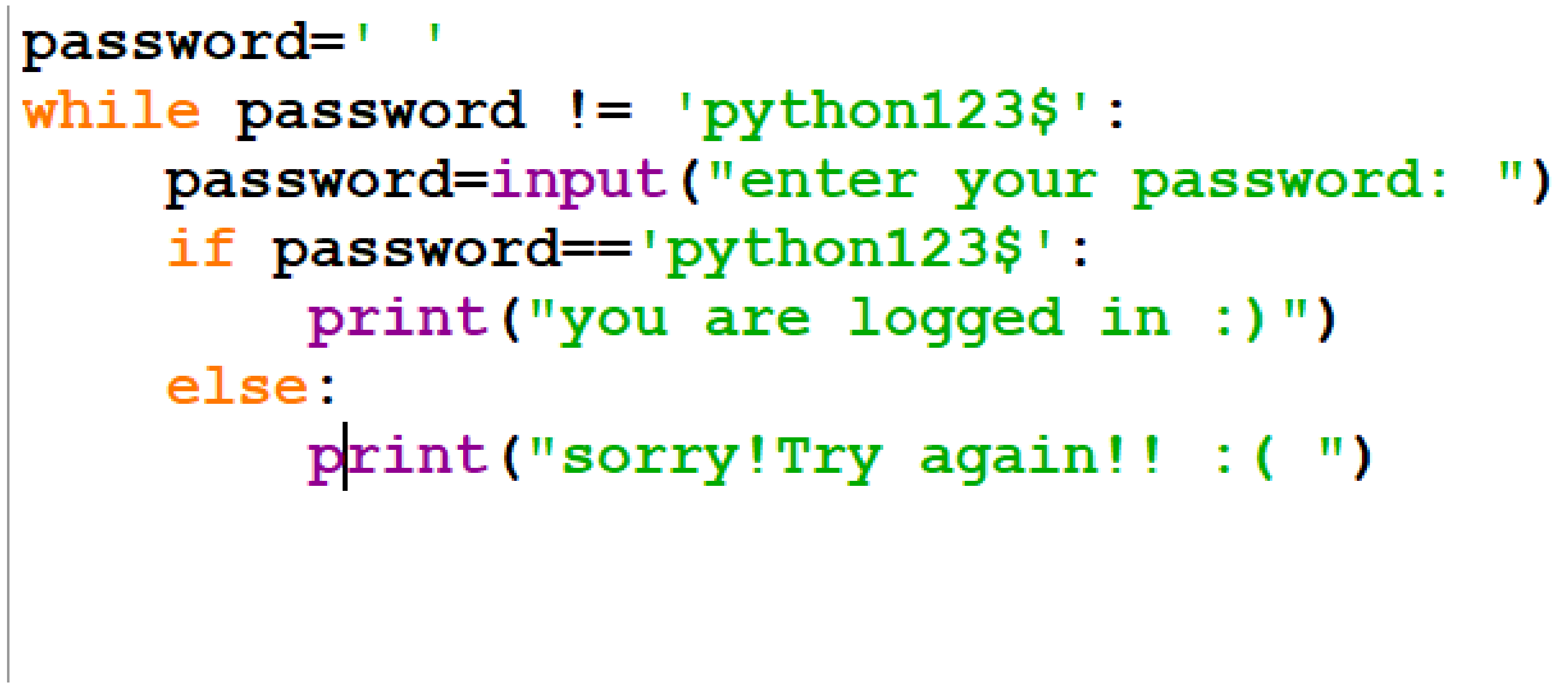
Example 1

The while loop statement becomes true so it enter to the loop it keep in executing because we didn’t write the script to come out of the loop



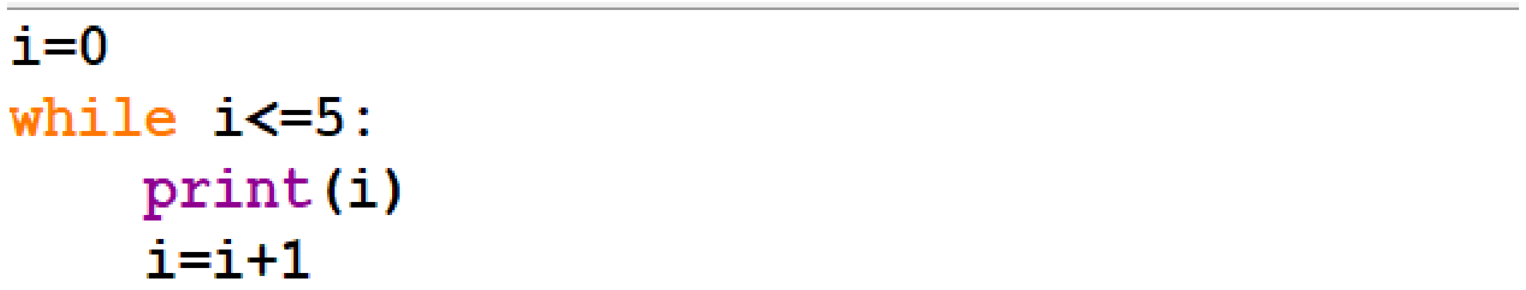
```
i=0
while i<=0:
    print(i)
```

Example 2:



```python
password=' '
while password != 'python123$':
    password=input("enter your password: ")
    if password=='python123$':
        print("you are logged in :)")
    else:
        print("sorry!Try again!! :( ")
```

Example 3

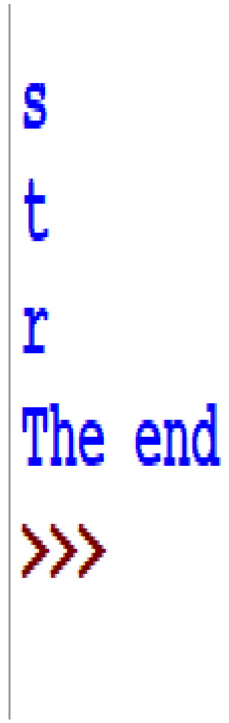


```
i=0
while i<=5:
    print(i)
    i=i+1
```

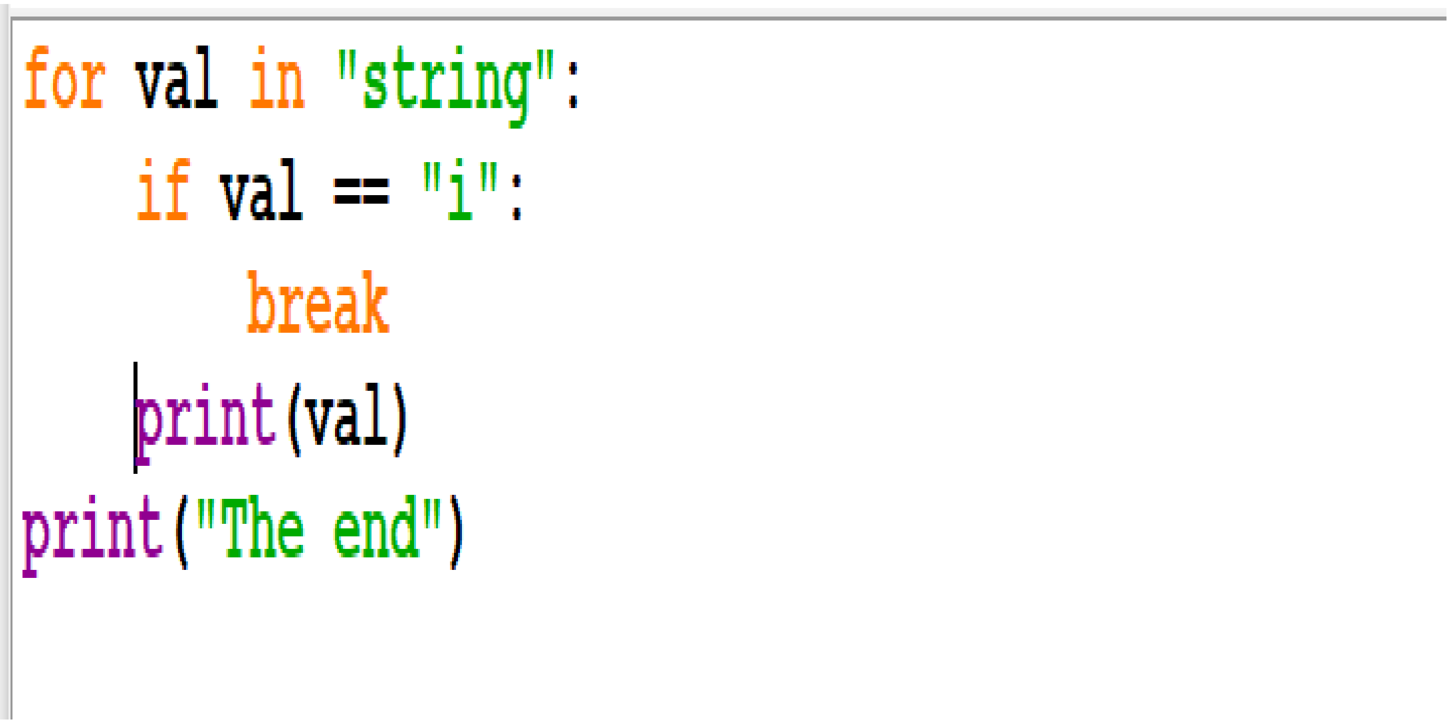
Use of Break and Continue Statement

- The ***break statement*** is used to exit the current loop (while and for loops). The scope of break statement is only the current loop.
- you need to exit just the current iteration of the loop rather exiting the loop entirely, you may use the **continue** statement of Python.

Example for Break Statement



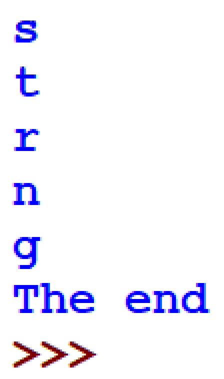
```
s
t
r
The end
>>>
```



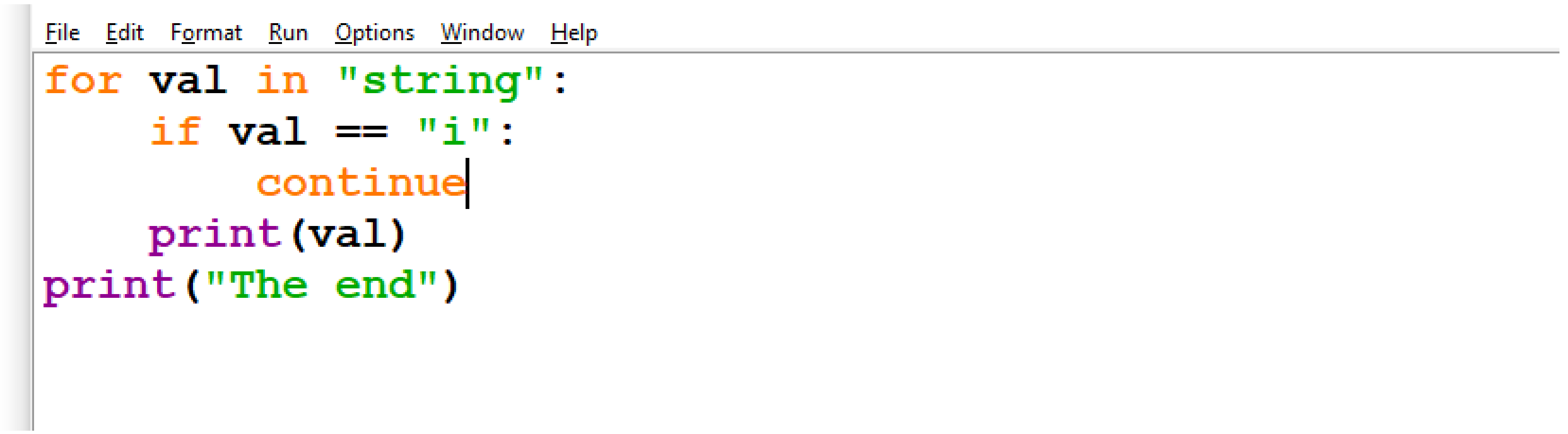
```
for val in "string":
    if val == "i":
        break
    print(val)
print("The end")
```

Continue statement

- continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.
- Syntax: continue



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



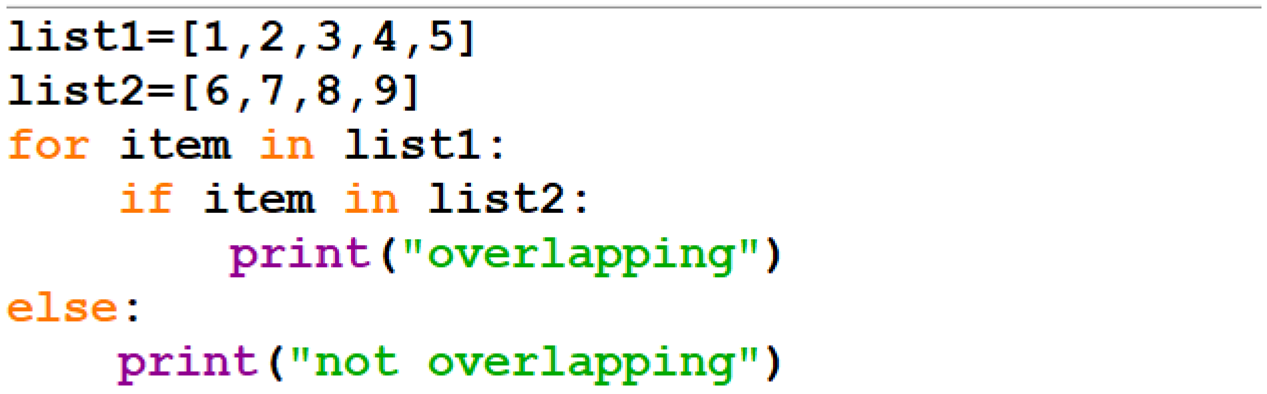
```
for val in "string":
    if val == "i":
        continue
    print(val)
print("The end")
```

Membership and Identity Operator

Example

In Operator Example

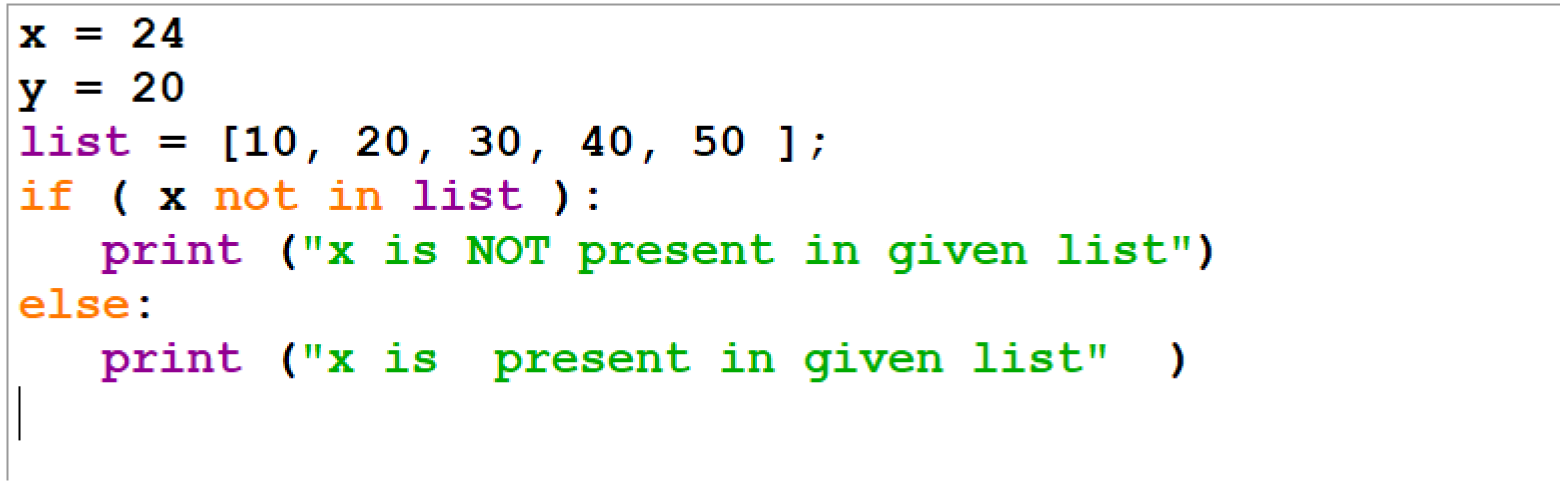
- **in operator :** The ‘in’ operator is used to check if a value exists in a sequence or not. Evaluates to true if it finds a variable in the specified sequence and false otherwise.
- Example:



```
list1=[1,2,3,4,5]
list2=[6,7,8,9]
for item in list1:
    if item in list2:
        print("overlapping")
else:
    print("not overlapping")
```

Not in operator Example

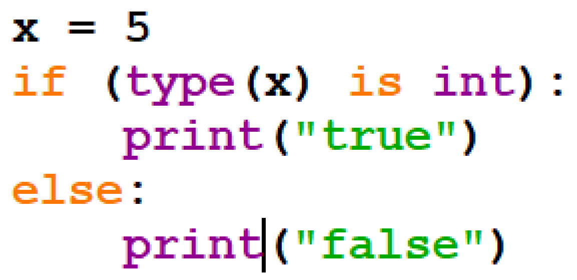
- **‘not in’ operator-** Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.



```
x = 24
y = 20
list = [10, 20, 30, 40, 50 ];
if ( x not in list ):
   print ("x is NOT present in given list")
else:
   print ("x is  present in given list"  )
```

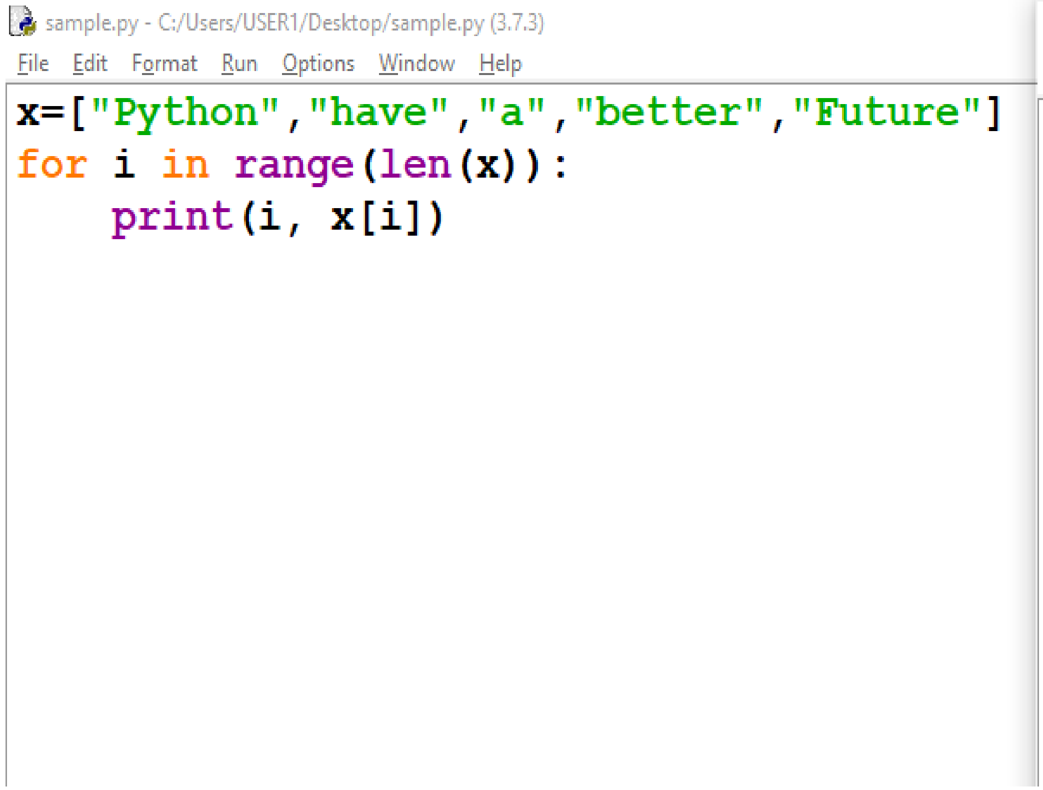
Is Operator

- **‘is’ operator –** Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.

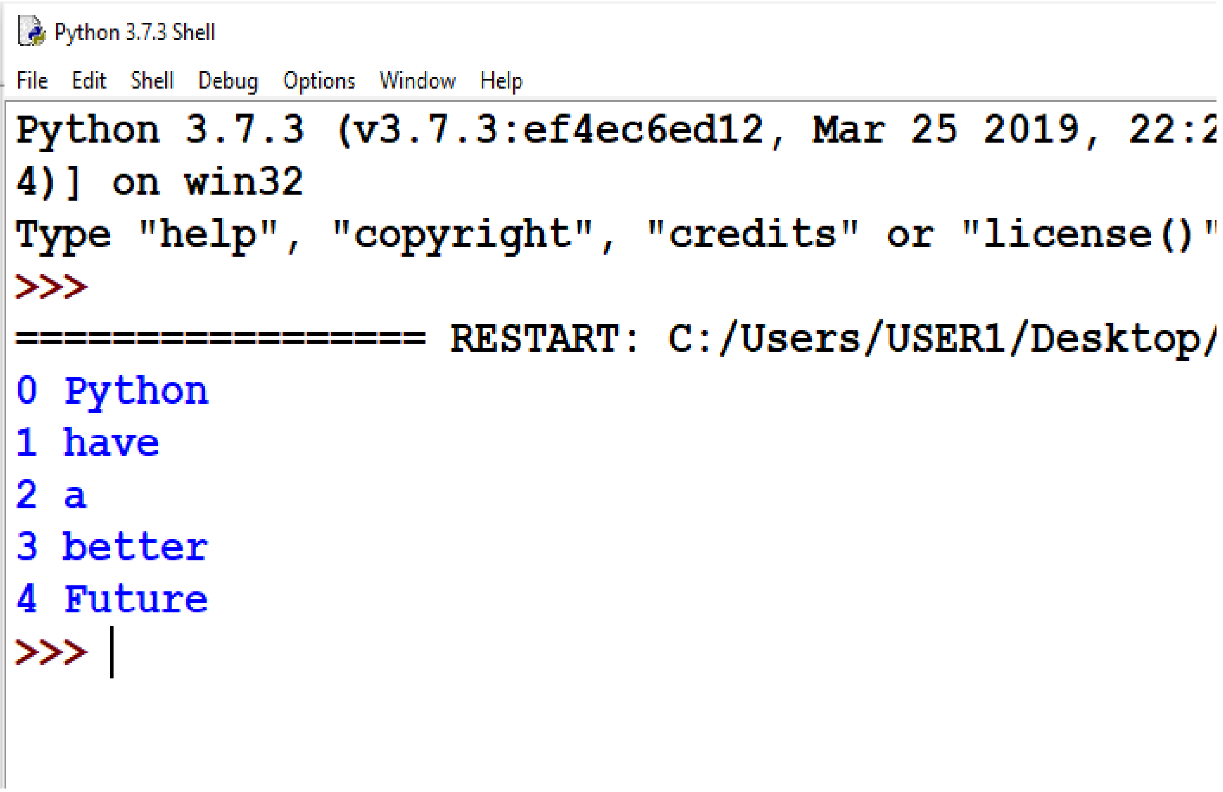


```
x = 5
if (type(x) is int):
    print("true")
else:
    print("false")
```

iterate over the indices of a sequence, you can combine range() and len() as follows:


sample.py - C:/Users/USER1/Desktop/sample.py (3.7.3)
File Edit Format Run Options Window Help

```
x=["Python","have","a","better","Future"]
for i in range(len(x)):
    print(i, x[i])
```



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:2
4)] on win32
Type "help", "copyright", "credits" or "license()"
>>>
================= RESTART: C:/Users/USER1/Desktop/
0 Python
1 have
2 a
3 better
4 Future
>>> |
```

Nested for loop

```
my_movies = [['How I Met Your Mother', 'Friends', 'Silicon Valley'],
    ['Family Guy', 'South Park', 'Rick and Morty'],
    ['Breaking Bad', 'Game of Thrones', 'The Wire']]
for sublist in my_movies:
    for movie_name in sublist:
        char_num = len(movie_name)
        print("The title " + movie_name + " is " + str(char_num) + " characters long.")
```

Good Job

Tasks