RV College of Engineering

INTRODUCTION TO PYTHON PROGRAMMING

UNIT III

For loops

- A for loop is used for iterating over a particular sequence (can be a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages
- Similar to an iterator method as found in other object-orientated programming languages.
- With the for loop a set of statements, once for each item in a list, tuple, set etc can be executed.
- Used for sequential traversal.

A simple ‘for’ loop

Syntax:

```
for iterator_var in sequence:
        statements(s)
```

Example 1: **output: 0 1 2 3**

```
n = 4
for i in range(0, n):
    print(i)
```

Example 2: **output: 4 3 2 1 stop!**

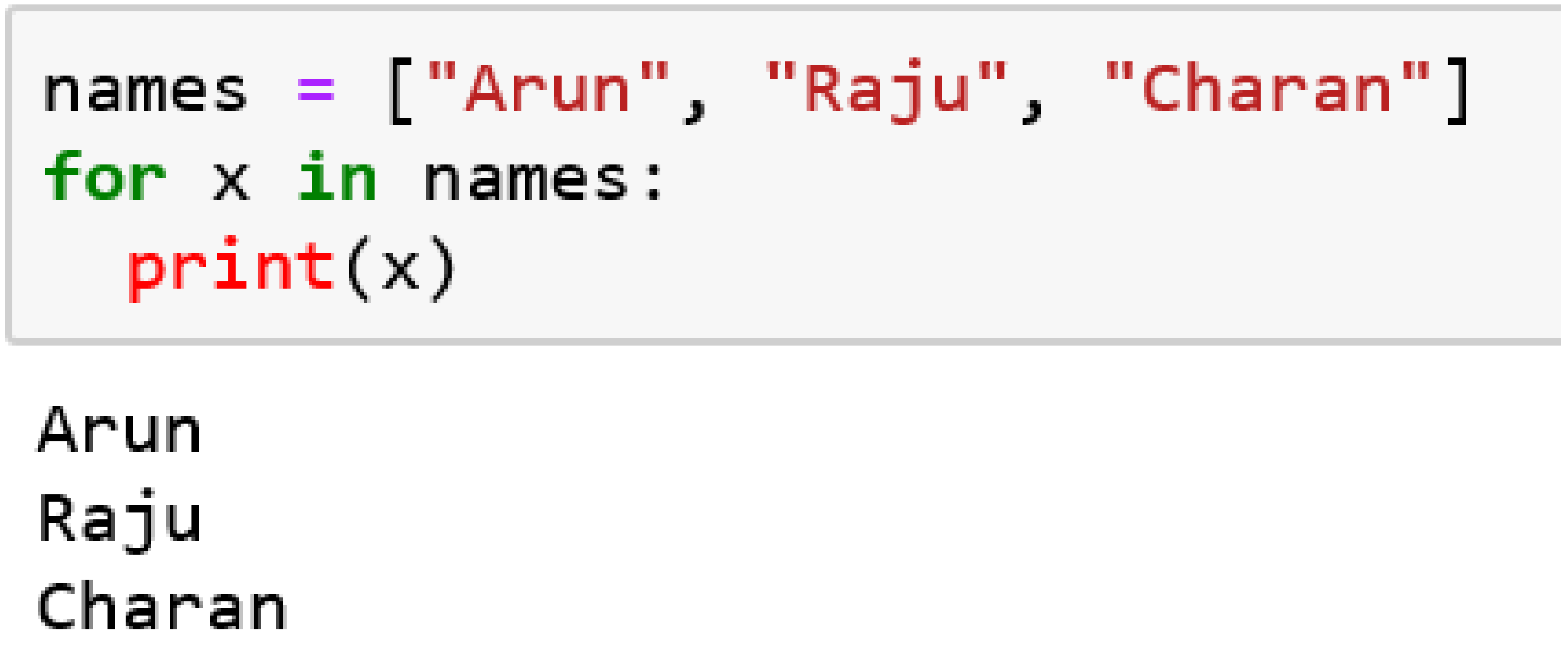
```
for i in [4, 3, 2, 1] :
    print(i)
print(‘stop!')
```

Example 3

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
   print(‘Hello:', friend)
print('Done!’)
```

```
Output:
Hello: Joseph
Hello: Glenn
Hello: Sally

Done!
```

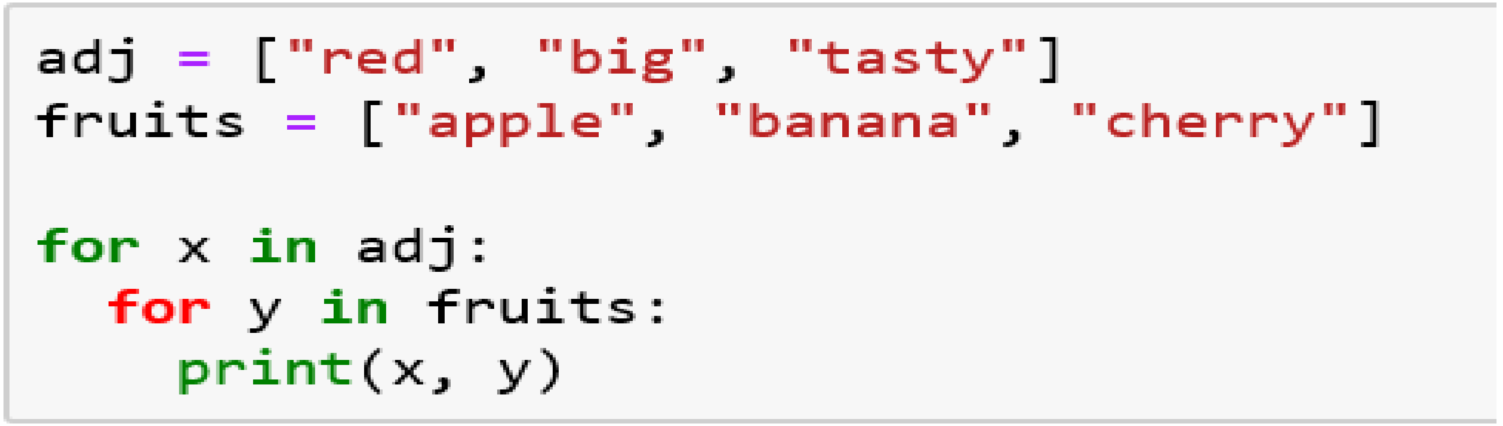
Example 4



```
names = ["Arun", "Raju", "Charan"]
for x in names:
  print(x)
```

```
Arun
Raju
Charan
```

Nested Loops



```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

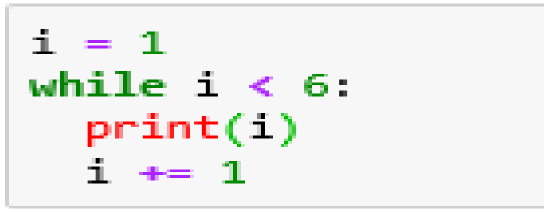
```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Control Statements

- Break
- Continue
- Pass

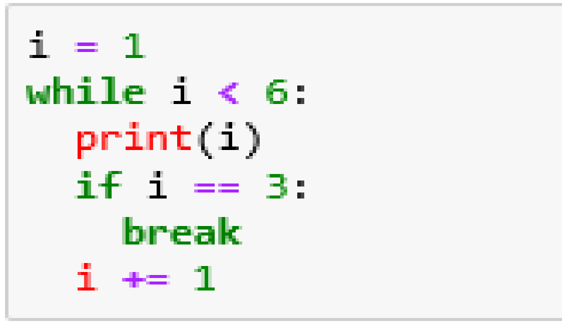
Break Statement

While Loop



```
i = 1
while i < 6:
  print(i)
  i += 1
```

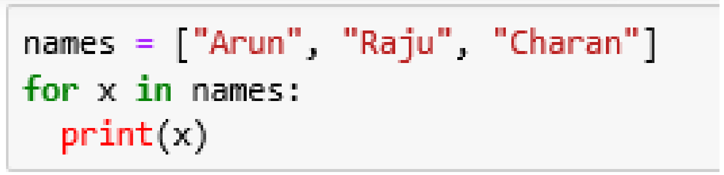
```
1
2
3
4
5
```



```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

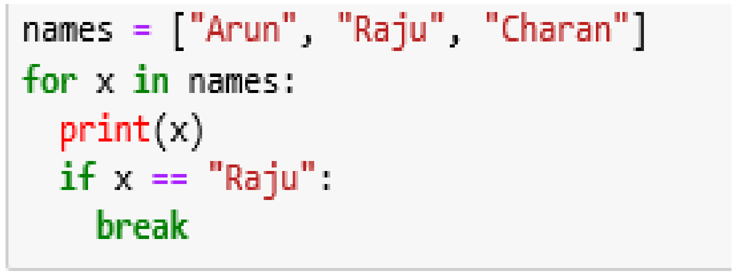
```
1
2
3
```

for Loop



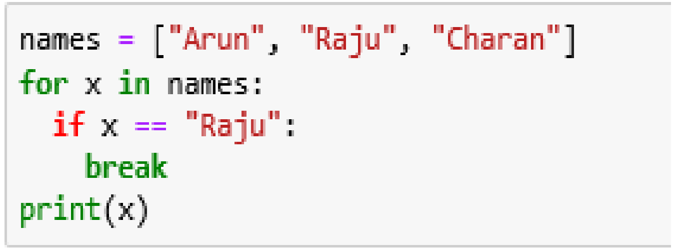
```
names = ["Arun", "Raju", "Charan"]
for x in names:
  print(x)
```

```
Arun
Raju
Charan
```



```
names = ["Arun", "Raju", "Charan"]
for x in names:
  print(x)
  if x == "Raju":
    break
```

```
Arun
Raju
```



```
names = ["Arun", "Raju", "Charan"]
for x in names:
  if x == "Raju":
    break
print(x)
```

```
Raju
```

Continue Statement

While Loop

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

for Loop

```
names = ["Arun", "Raju", "Charan"]
for x in names:
  print(x)
```

```
Arun
Raju
Charan
```

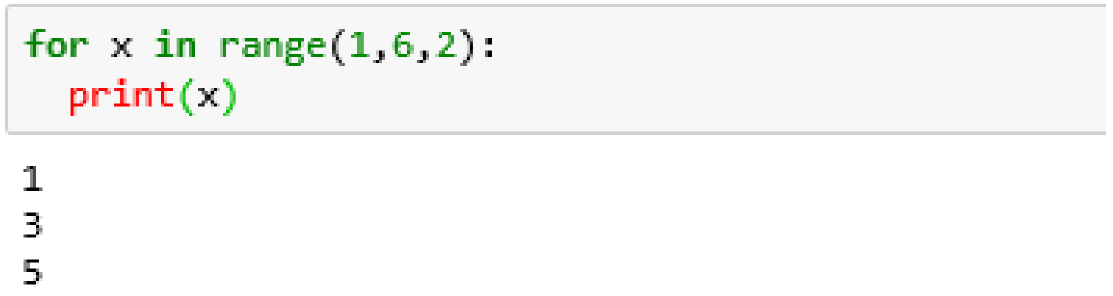
```
names = ["Arun", "Raju", "Charan"]
for x in names:
  if x == "Raju":
    continue
  print(x)
```

```
Arun
Charan
```

RV College of Engineering

Pass Statement

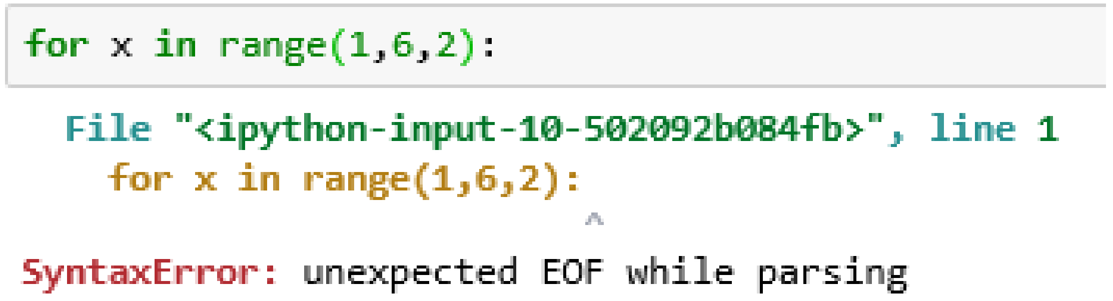
For loop



```
for x in range(1,6,2):
  print(x)
```

```
1
3
5
```

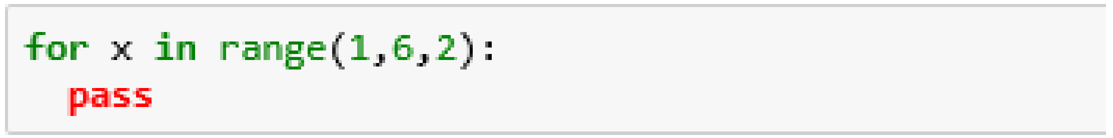
Empty For loop



```
for x in range(1,6,2):
```

```
  File "<ipython-input-10-502092b084fb>", line 1
    for x in range(1,6,2):
                          ^
SyntaxError: unexpected EOF while parsing
```

Empty For loop



```
for x in range(1,6,2):
  pass
```

Strings

- Accessing Strings
- Basic Operations
- String slices
- Function and Methods

Strings

- String Initialization

  a='Welcome, to RVCE!'

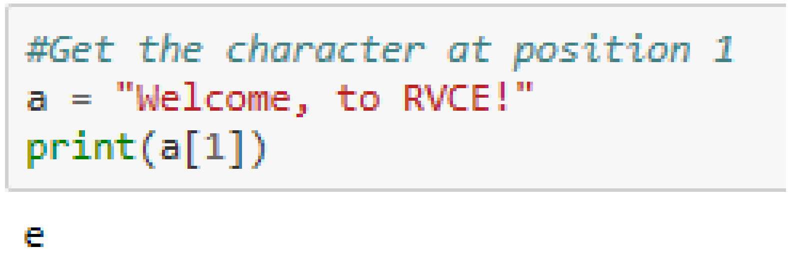
  b = "Welcome, to RVCE!"

  c="""Welcome,

  to RVCE!"""“

- Accessing characters in a string

  a[1]



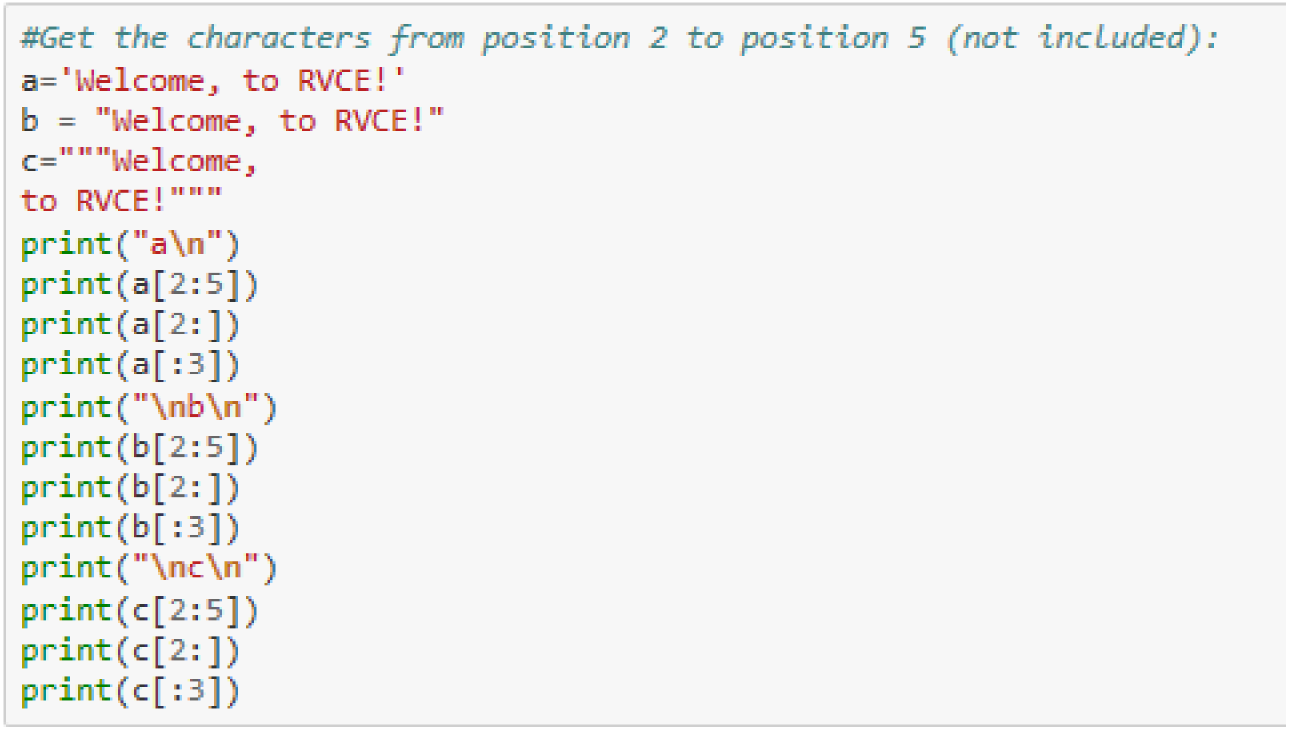
```
#Get the character at position 1
a = "Welcome, to RVCE!"
print(a[1])
```

```
e
```

Strings

Substring

a='Welcome, to RVCE!‘
Print(a)
print(a[2:5])
print(a[2:])
print(a[:3])



```
#Get the characters from position 2 to position 5 (not included):
a='Welcome, to RVCE!'
b = "Welcome, to RVCE!"
c="""Welcome,
to RVCE!"""
print("a\n")
print(a[2:5])
print(a[2:])
print(a[:3])
print("\nb\n")
print(b[2:5])
print(b[2:])
print(b[:3])
print("\nc\n")
print(c[2:5])
print(c[2:])
print(c[:3])
```

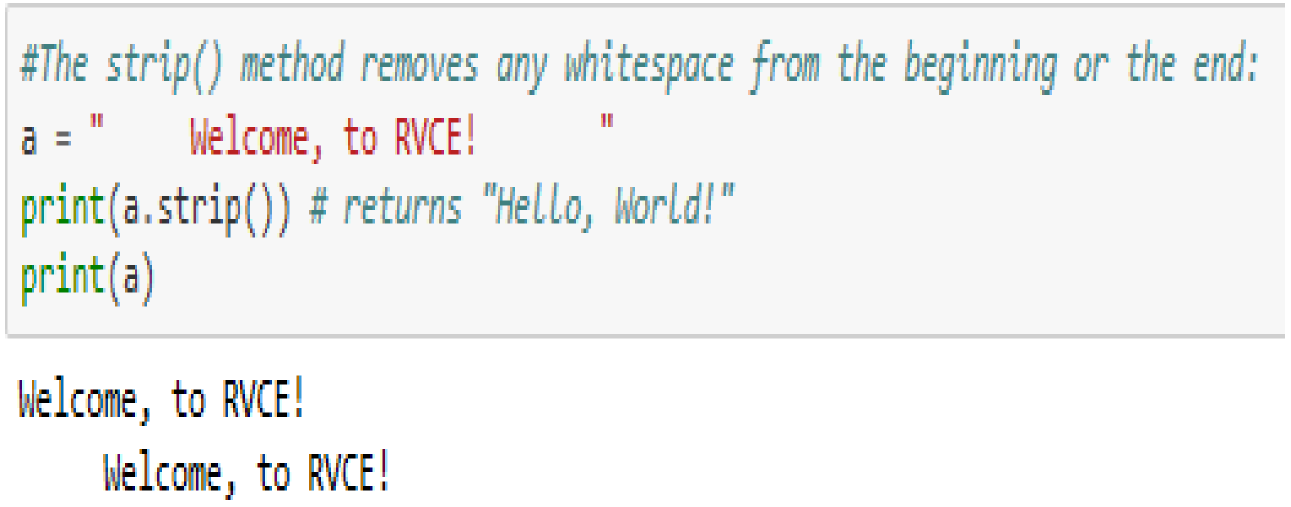
```
a

lco
lcome, to RVCE!
Wel

b

lco
lcome, to RVCE!
Wel

c

lco
lcome,
to RVCE!
Wel
```

String- In built functions

strip() -- removes whitespace from the beginning or end

a = “   Welcome, to RVCE!   “

print(a.strip())



```
#The strip() method removes any whitespace from the beginning or the end:
a = "     Welcome, to RVCE!       "
print(a.strip()) # returns "Hello, World!"
print(a)
```

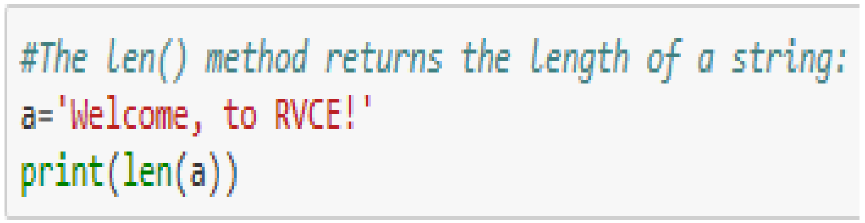
```
Welcome, to RVCE!
     Welcome, to RVCE!
```

String- In built functions

- len() -- returns the length of a string

a = “ Welcome, to RVCE! “

print(len(a))

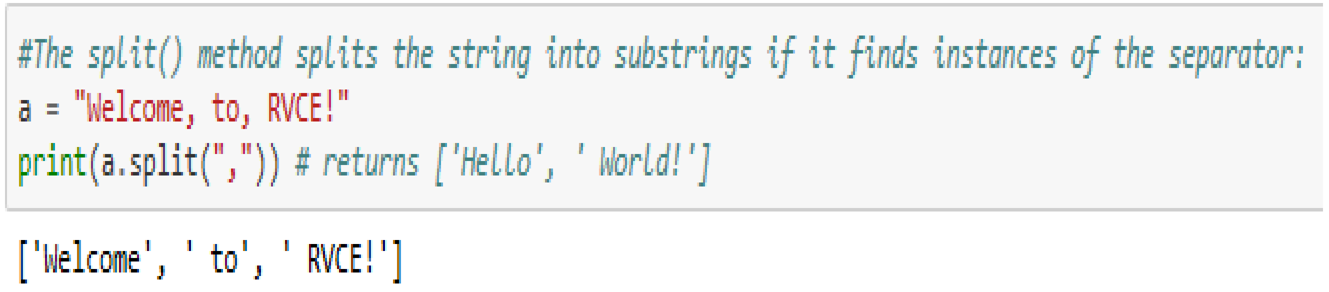


```
#The len() method returns the length of a string:
a='Welcome, to RVCE!'
print(len(a))
```

```
17
```

String- In built functions

- split() -- splits the string into substrings if it finds instances of the separator

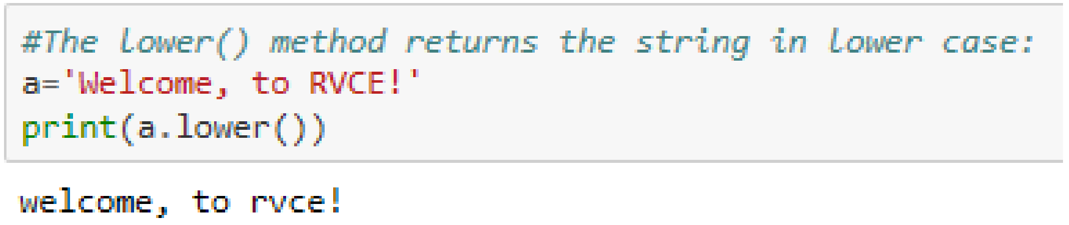


```
#The split() method splits the string into substrings if it finds instances of the separator:
a = "Welcome, to, RVCE!"
print(a.split(",")) # returns ['Hello', ' World!']
```

```
['Welcome', ' to', ' RVCE!']
```

String- In built functions

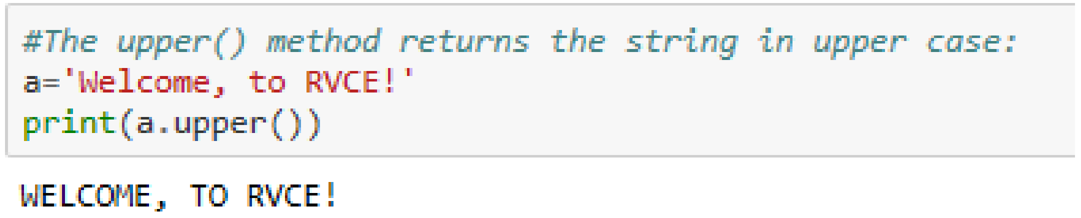
lower() -- returns the lower case of the string



```
#The lower() method returns the string in lower case:
a='Welcome, to RVCE!'
print(a.lower())
```

```
welcome, to rvce!
```

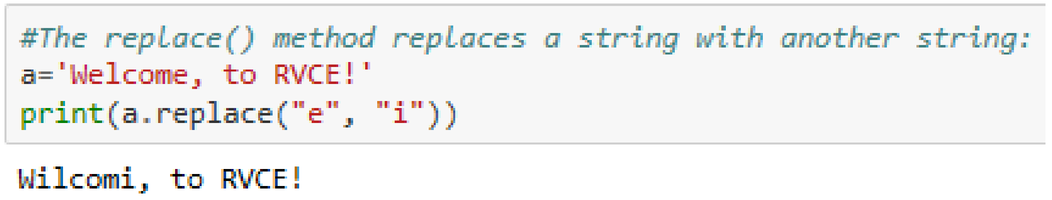
upper() -- returns the upper case of the string



```
#The upper() method returns the string in upper case:
a='Welcome, to RVCE!'
print(a.upper())
```

```
WELCOME, TO RVCE!
```

lower() -- returns the replaced string



```
#The replace() method replaces a string with another string:
a='Welcome, to RVCE!'
print(a.replace("e", "i"))
```

```
Wilcomi, to RVCE!
```

String manipulation functions

- capitalize() Capitalizes first letter of string
- center (width, fillchar) Returns a space-padded string with the original string centered to a total of width columns.
- count(str, beg= 0,end=len(string)) Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given
- endswith(suffix, beg=0, end=len(string)) Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise
- Find (str, beg=0 end=len(string))Determine if str occurs in string

String manipulation functions

- **isalnum()**Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
- **isdigit()**Returns true if string contains only digits and false otherwise.
- **islower()**Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
- **isspace()**Returns true if string contains only whitespace characters and false otherwise.

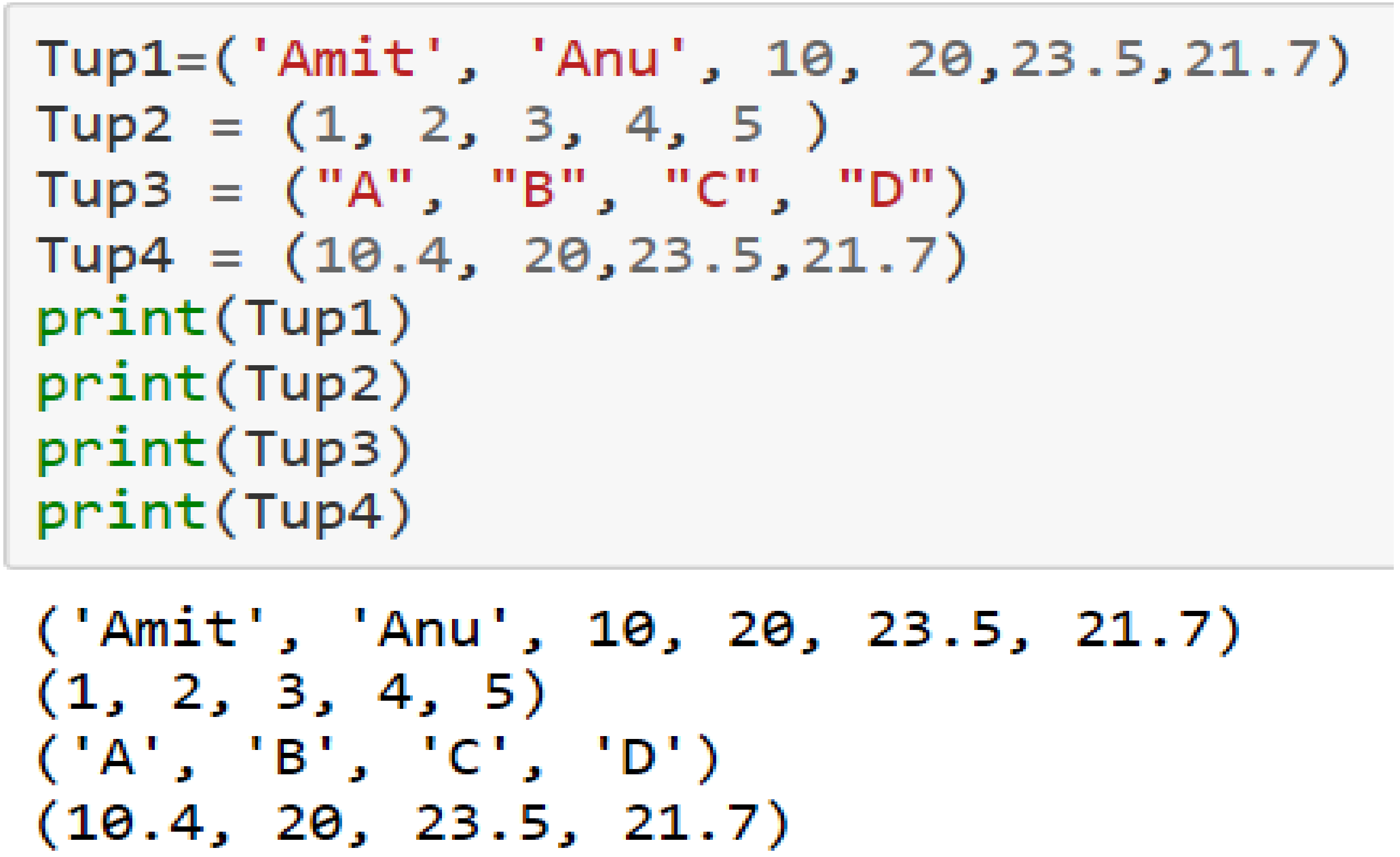
String Formatting Operator

- %c for character
- %s for strings
- %o for octal numbers
- %x for hexadecimal numbers
- %f for floating point numbers

Tuples in Python

- A tuple is a collection which is ordered and **unchangeable**.
- Tuples are written with round brackets.

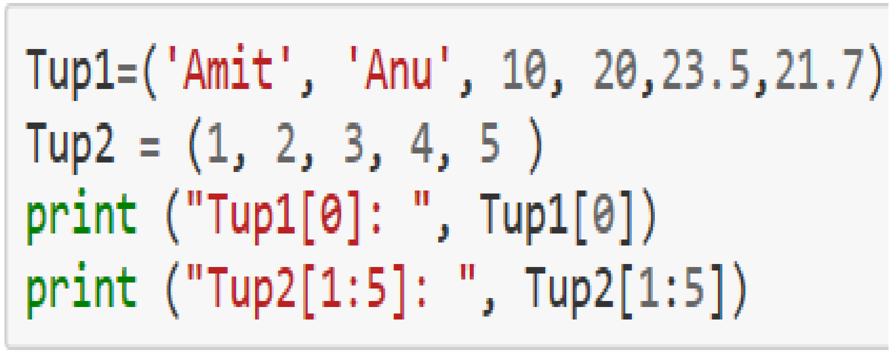
Creating tuple



```
Tup1=('Amit', 'Anu', 10, 20,23.5,21.7)
Tup2 = (1, 2, 3, 4, 5 )
Tup3 = ("A", "B", "C", "D")
Tup4 = (10.4, 20,23.5,21.7)
print(Tup1)
print(Tup2)
print(Tup3)
print(Tup4)
```

```
('Amit', 'Anu', 10, 20, 23.5, 21.7)
(1, 2, 3, 4, 5)
('A', 'B', 'C', 'D')
(10.4, 20, 23.5, 21.7)
```

Accessing Values in Tuple



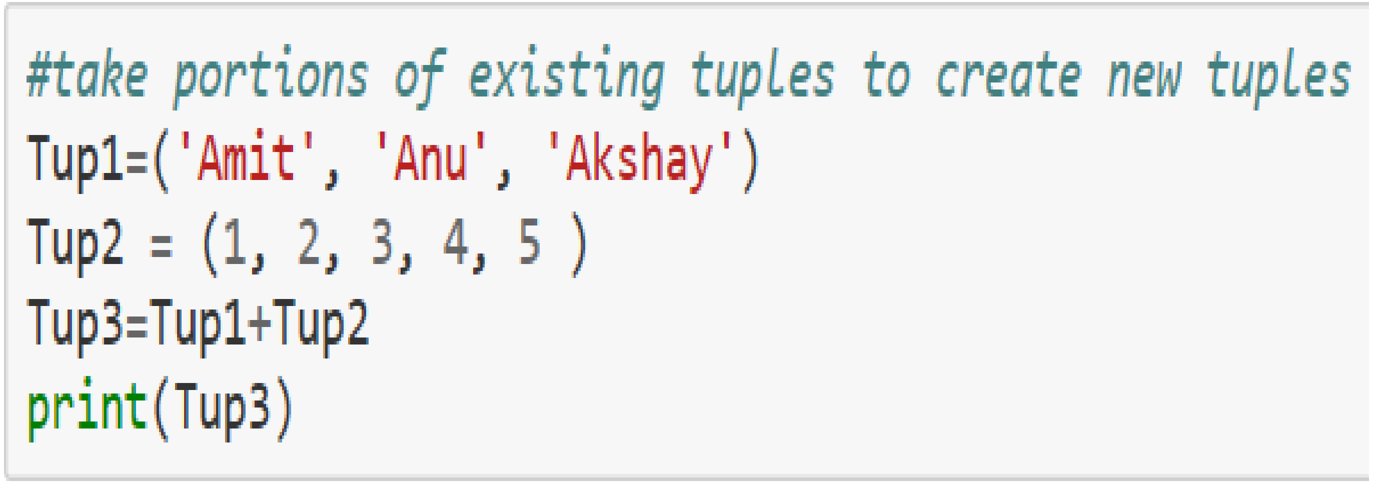
```
Tup1=('Amit', 'Anu', 10, 20,23.5,21.7)
Tup2 = (1, 2, 3, 4, 5 )
print ("Tup1[0]: ", Tup1[0])
print ("Tup2[1:5]: ", Tup2[1:5])
```

```
Tup1[0]:  Amit
Tup2[1:5]:  (2, 3, 4, 5)
```

Updating values in Tuple

- Tuples are immutable which means you cannot update or change the values of tuple elements

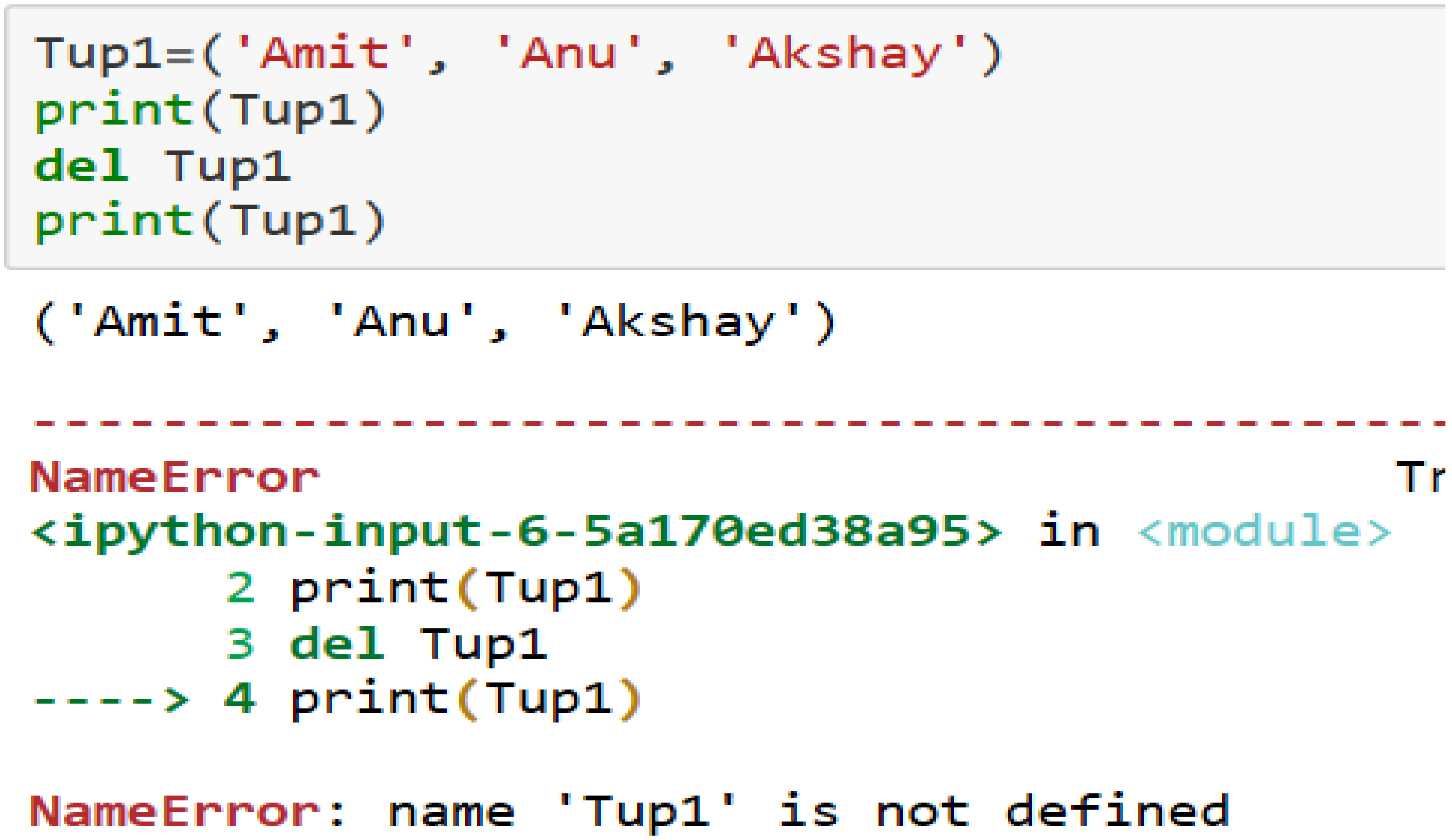
Updating values in Tuple



```
#take portions of existing tuples to create new tuples
Tup1=('Amit', 'Anu', 'Akshay')
Tup2 = (1, 2, 3, 4, 5 )
Tup3=Tup1+Tup2
print(Tup3)
```

```
('Amit', 'Anu', 'Akshay', 1, 2, 3, 4, 5)
```

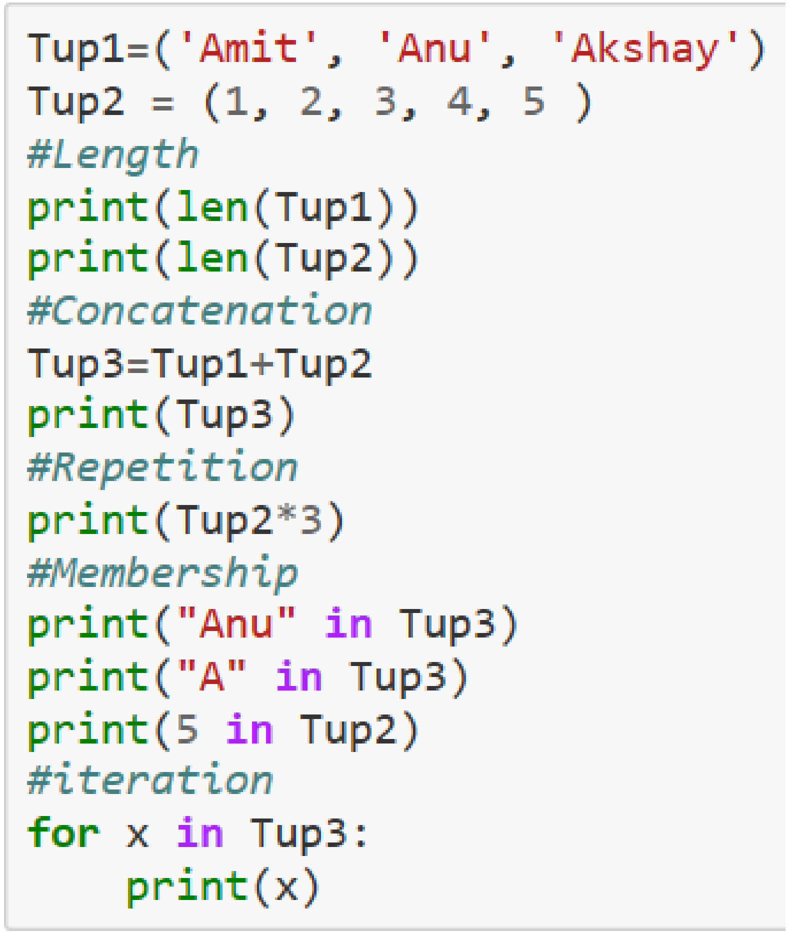
Delete Tuple Elements



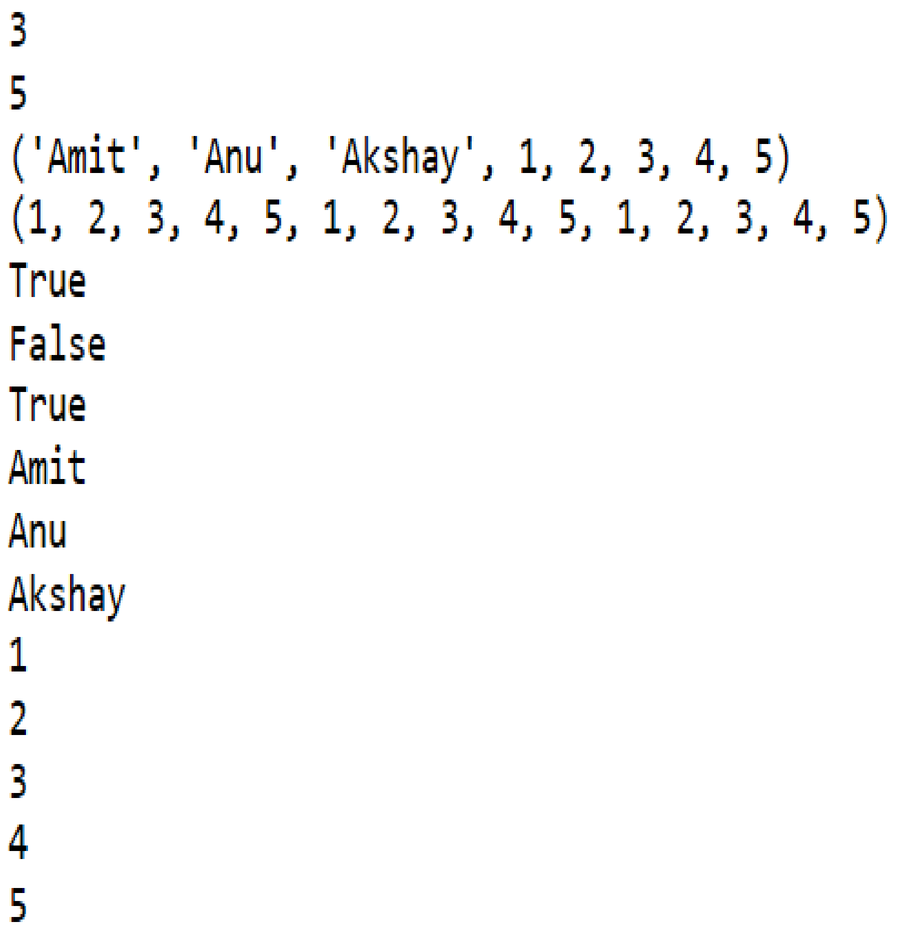
```
Tup1=('Amit', 'Anu', 'Akshay')
print(Tup1)
del Tup1
print(Tup1)
```

```
('Amit', 'Anu', 'Akshay')

---------------------------------------------------------------------------
NameError                                 Tr
<ipython-input-6-5a170ed38a95> in <module>
      2 print(Tup1)
      3 del Tup1
----> 4 print(Tup1)

NameError: name 'Tup1' is not defined
```

Basic Tuples Operations

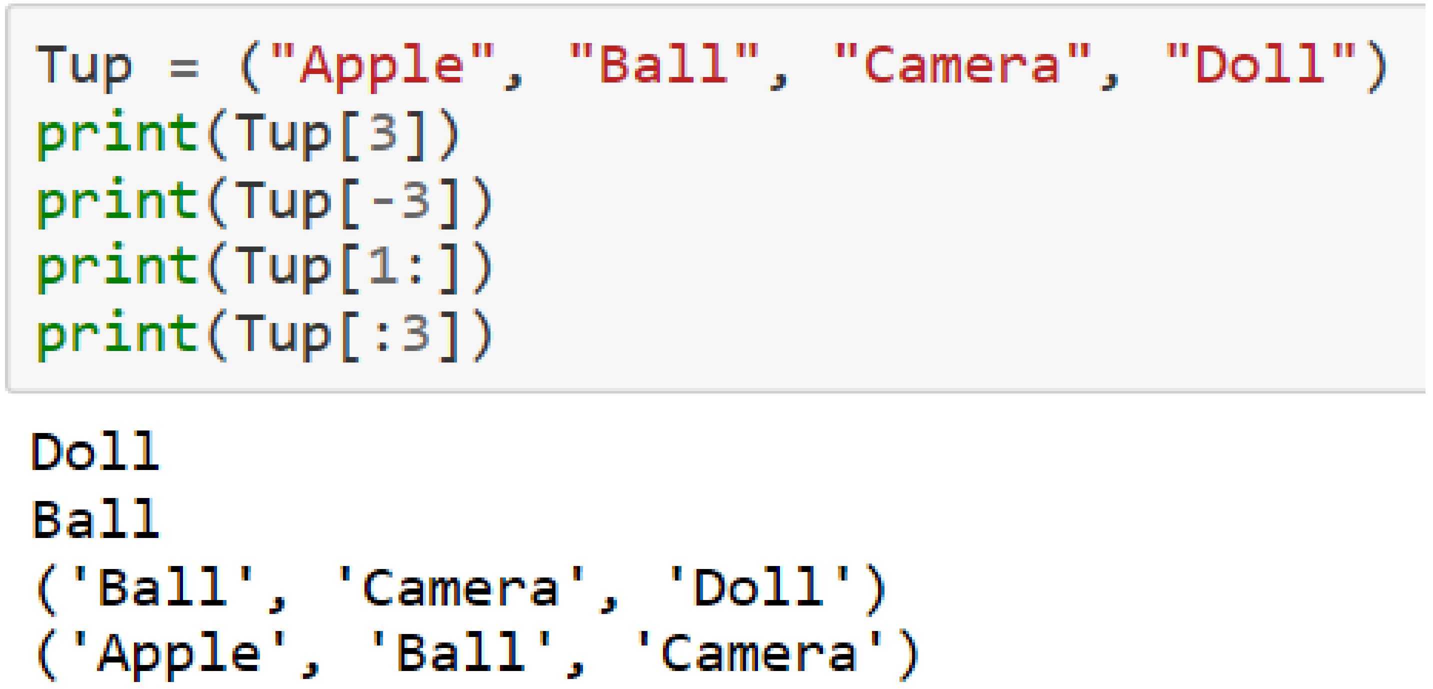


```
Tup1=('Amit', 'Anu', 'Akshay')
Tup2 = (1, 2, 3, 4, 5 )
#Length
print(len(Tup1))
print(len(Tup2))
#Concatenation
Tup3=Tup1+Tup2
print(Tup3)
#Repetition
print(Tup2*3)
#Membership
print("Anu" in Tup3)
print("A" in Tup3)
print(5 in Tup2)
#iteration
for x in Tup3:
    print(x)
```



```
3
5
('Amit', 'Anu', 'Akshay', 1, 2, 3, 4, 5)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
True
False
True
Amit
Anu
Akshay
1
2
3
4
5
```

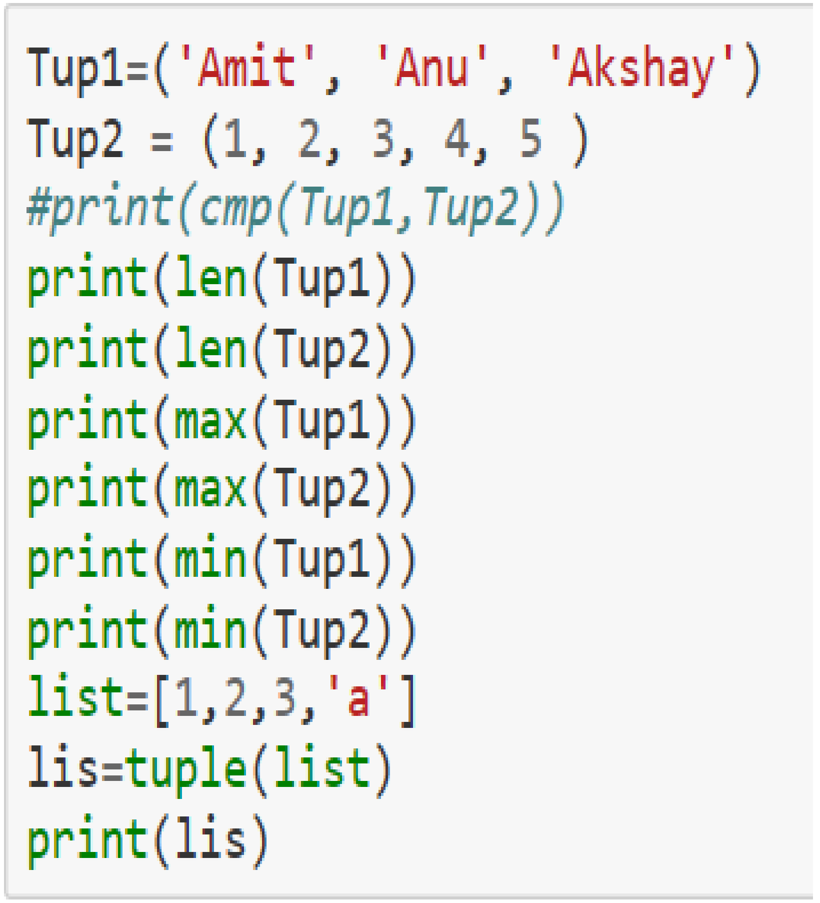
Indexing, Slicing, and Matrixes



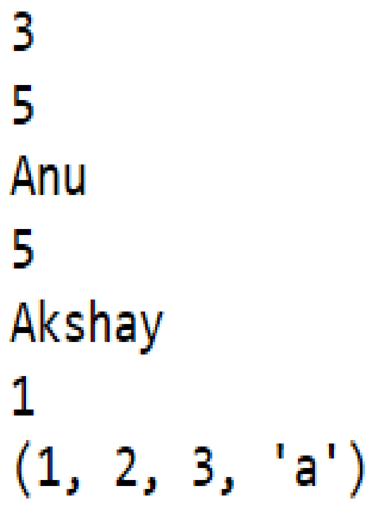
```python
Tup = ("Apple", "Ball", "Camera", "Doll")
print(Tup[3])
print(Tup[-3])
print(Tup[1:])
print(Tup[:3])
```

```
Doll
Ball
('Ball', 'Camera', 'Doll')
('Apple', 'Ball', 'Camera')
```

Built-in Tuple Functions



```
Tup1=('Amit', 'Anu', 'Akshay')
Tup2 = (1, 2, 3, 4, 5 )
#print(cmp(Tup1,Tup2))
print(len(Tup1))
print(len(Tup2))
print(max(Tup1))
print(max(Tup2))
print(min(Tup1))
print(min(Tup2))
list=[1,2,3,'a']
lis=tuple(list)
print(lis)
```



```
3
5
Anu
5
Akshay
1
(1, 2, 3, 'a')
```

Lists in Python

- Lists are used to store multiple items using a single variable.
- Lists is a built-in data types
- Lists are used to store collections of data,
- Similar to lists are Tuple, Set, and Dictionary
- List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1] and so on
- List items can be of any data type

Lists in Python

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john’]
print (list)                # Prints complete list
print (list[0])             # Prints first element of the list
print (list[1:3])           # Prints elements starting from 2nd till 3rd
print (list[2:])            # Prints elements starting from 3rd element
print (tinylist  * 2)       # Prints list two times
print (list + tinylist)     # Prints concatenated lists
```

Lists in Python

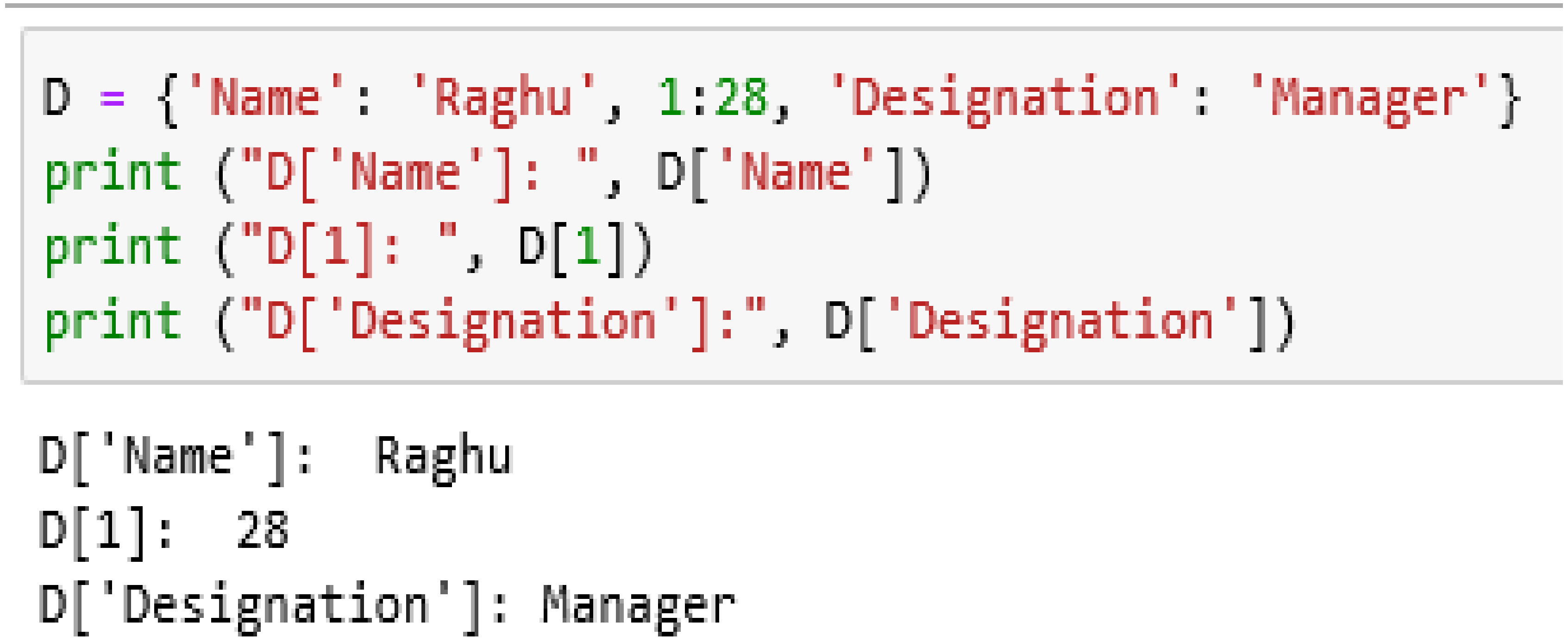
Output:

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john’]
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

Dictionary

- Key is separated from its value by a colon (:)
- Items are separated by commas
- Dictionary is enclosed in curly braces
- Keys are unique within a dictionary while values may not be.

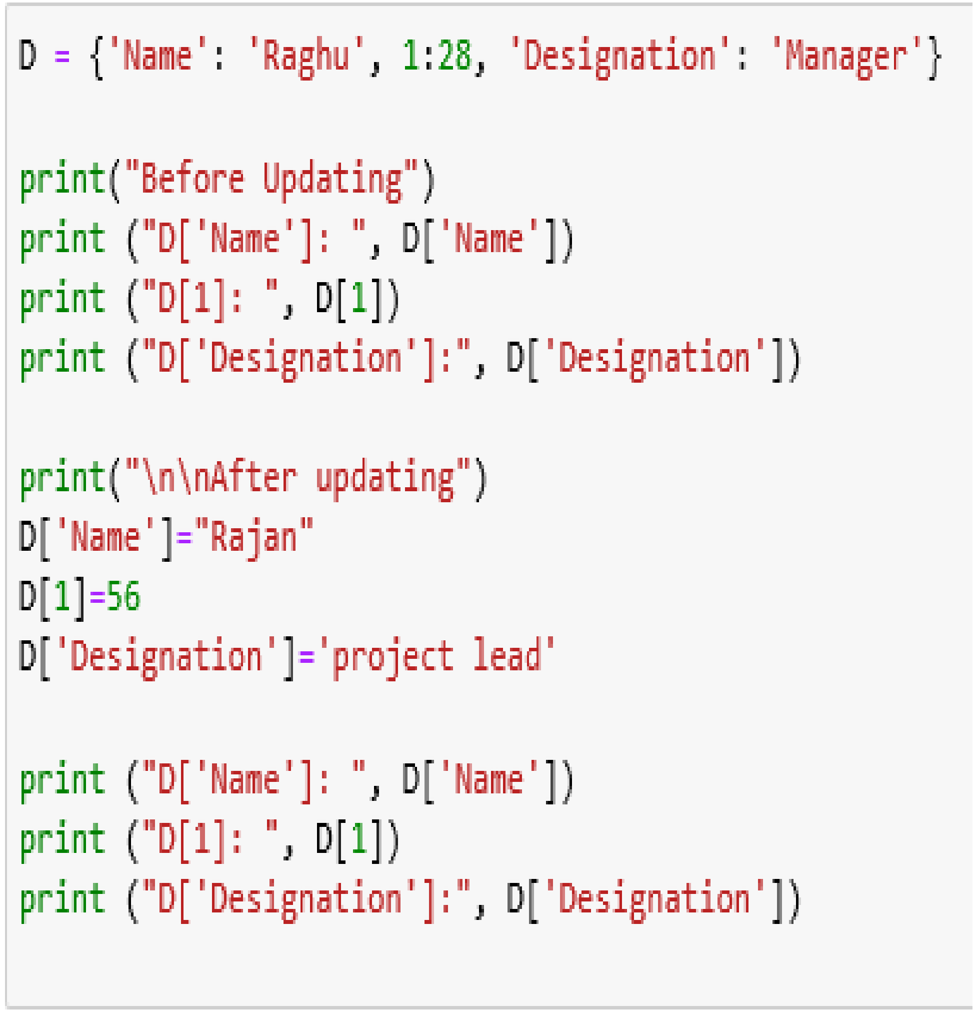
Accessing Values in Dictionary



```
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print ("D['Name']: ", D['Name'])
print ("D[1]: ", D[1])
print ("D['Designation']:", D['Designation'])
```

```
D['Name']:  Raghu
D[1]:  28
D['Designation']: Manager
```

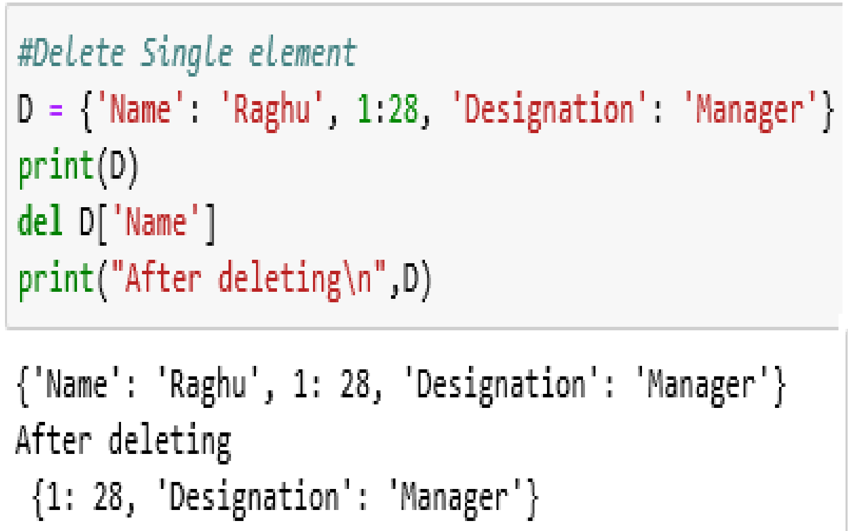
Updating Dictionary



```python
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}

print("Before Updating")
print ("D['Name']: ", D['Name'])
print ("D[1]: ", D[1])
print ("D['Designation']:", D['Designation'])

print("\n\nAfter updating")
D['Name']="Rajan"
D[1]=56
D['Designation']='project lead'

print ("D['Name']: ", D['Name'])
print ("D[1]: ", D[1])
print ("D['Designation']:", D['Designation'])
```

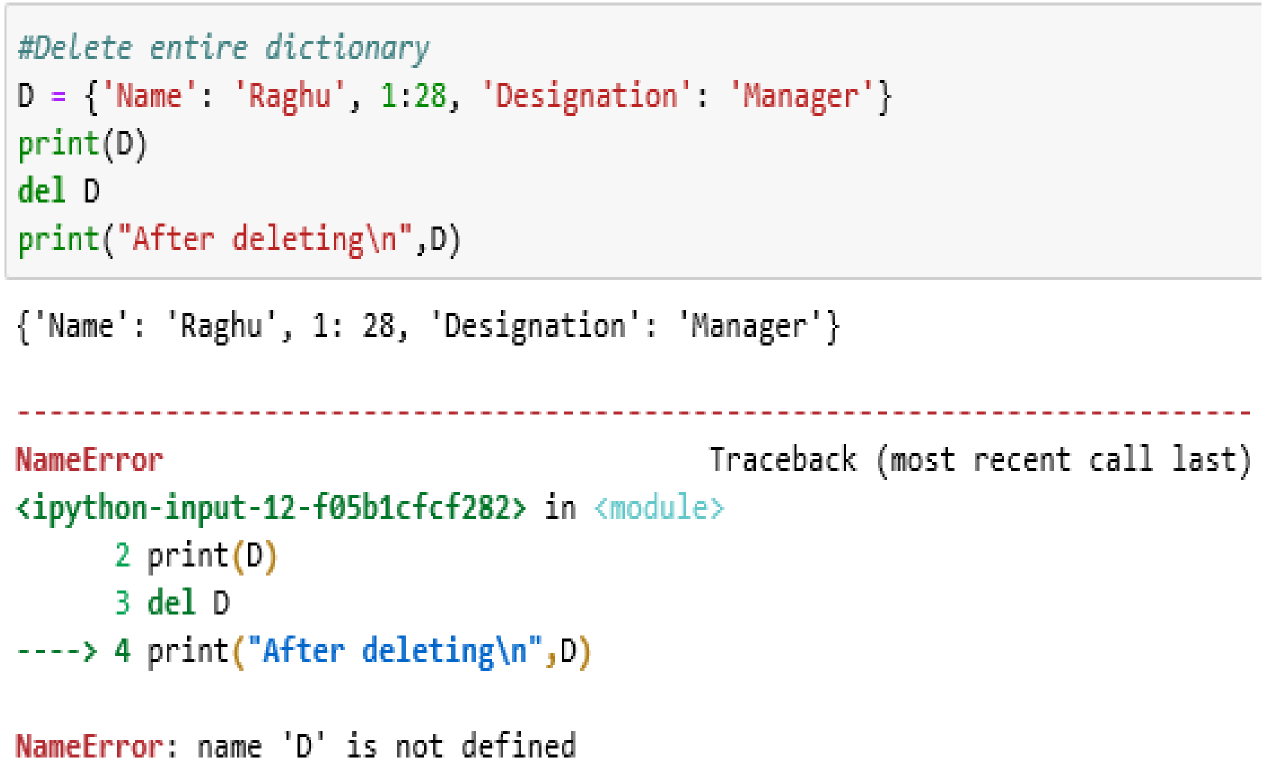
```
Before Updating
D['Name']:  Raghu
D[1]:  28
D['Designation']: Manager


After updating
D['Name']:  Rajan
D[1]:  56
D['Designation']: project lead
```

Delete Dictionary Elements



```
#Delete Single element
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print(D)
del D['Name']
print("After deleting\n",D)
```

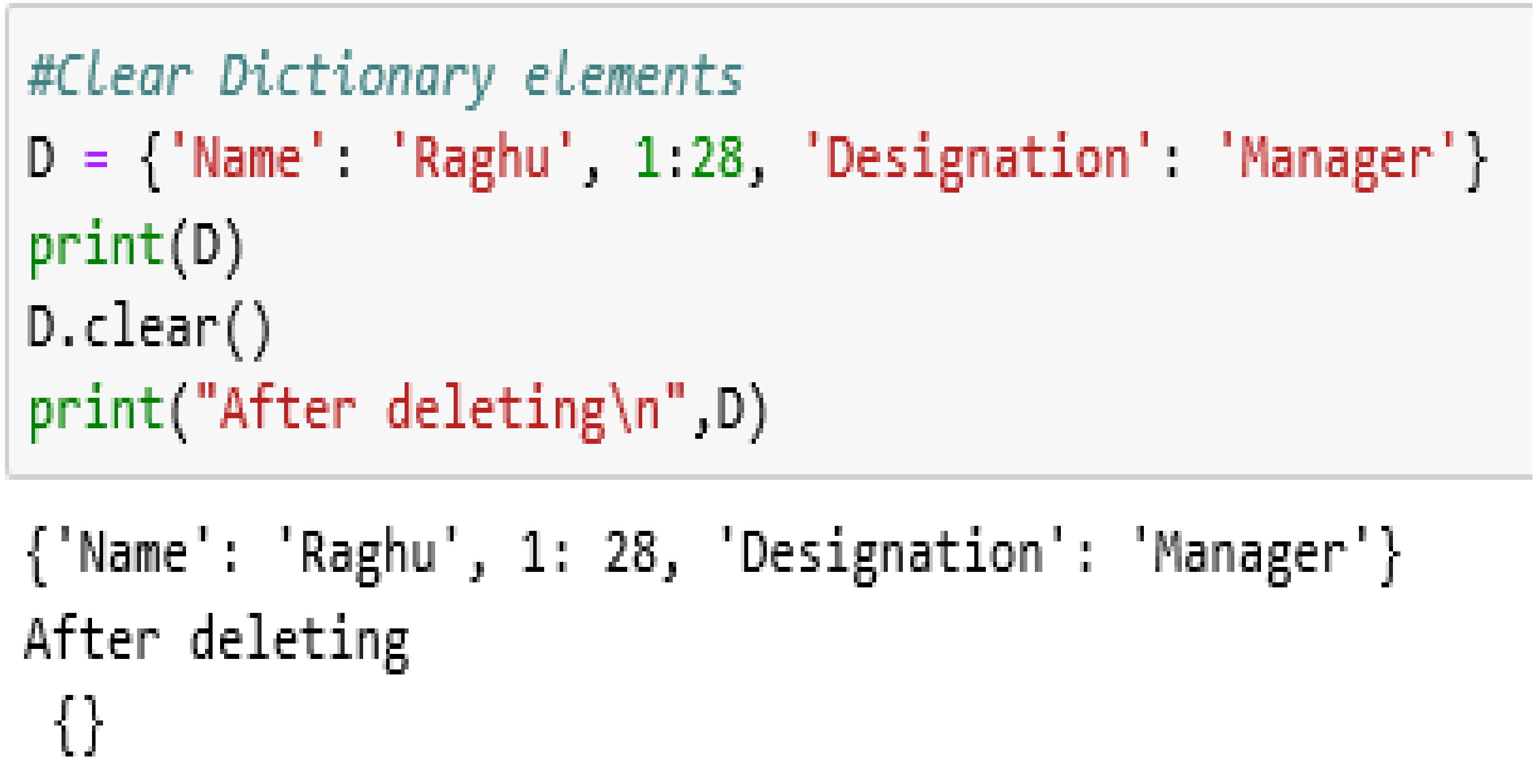
```
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
After deleting
 {1: 28, 'Designation': 'Manager'}
```



```
#Delete entire dictionary
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print(D)
del D
print("After deleting\n",D)
```

```
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-f05b1cfcf282> in <module>
      2 print(D)
      3 del D
----> 4 print("After deleting\n",D)

NameError: name 'D' is not defined
```

Clear Dictionary elements



```python
#Clear Dictionary elements
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print(D)
D.clear()
print("After deleting\n",D)
```

```
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
After deleting
 {}
```

Properties of Dictionary Keys

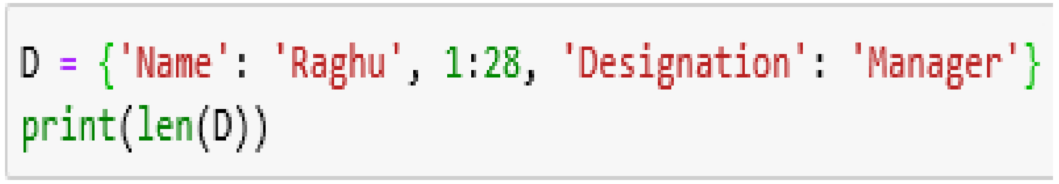
(a) Only one entry per key, if duplicate key is used then the last assignment wins.

(b) Keys must be immutable

RV College of Engineering

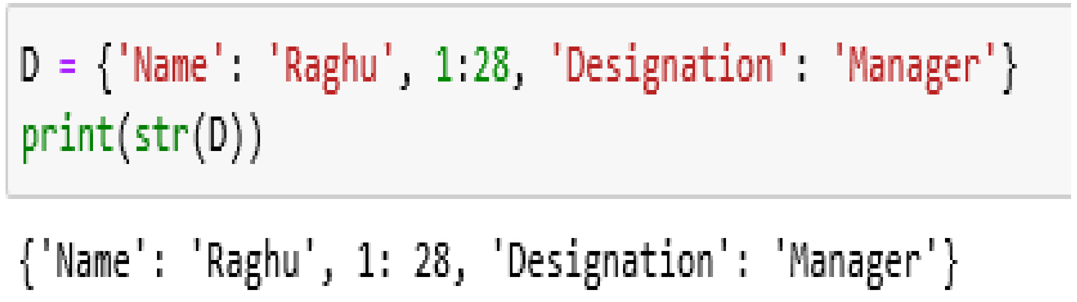
Built-in Dictionary Functions

- len(dict) Gives the total length of the dictionary



```
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print(len(D))
```

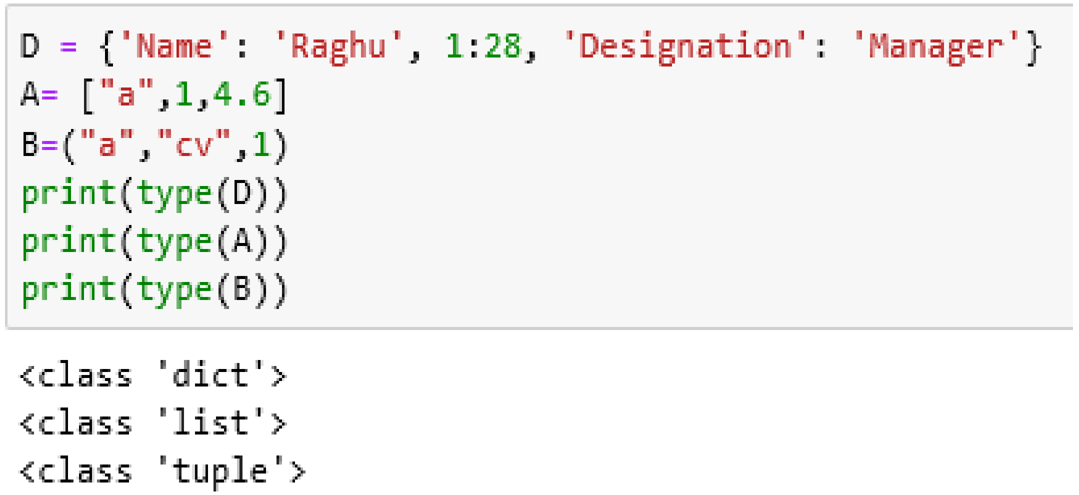
- str(dict) Produces a string representation of dictionary



```
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print(str(D))
```

```
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
```

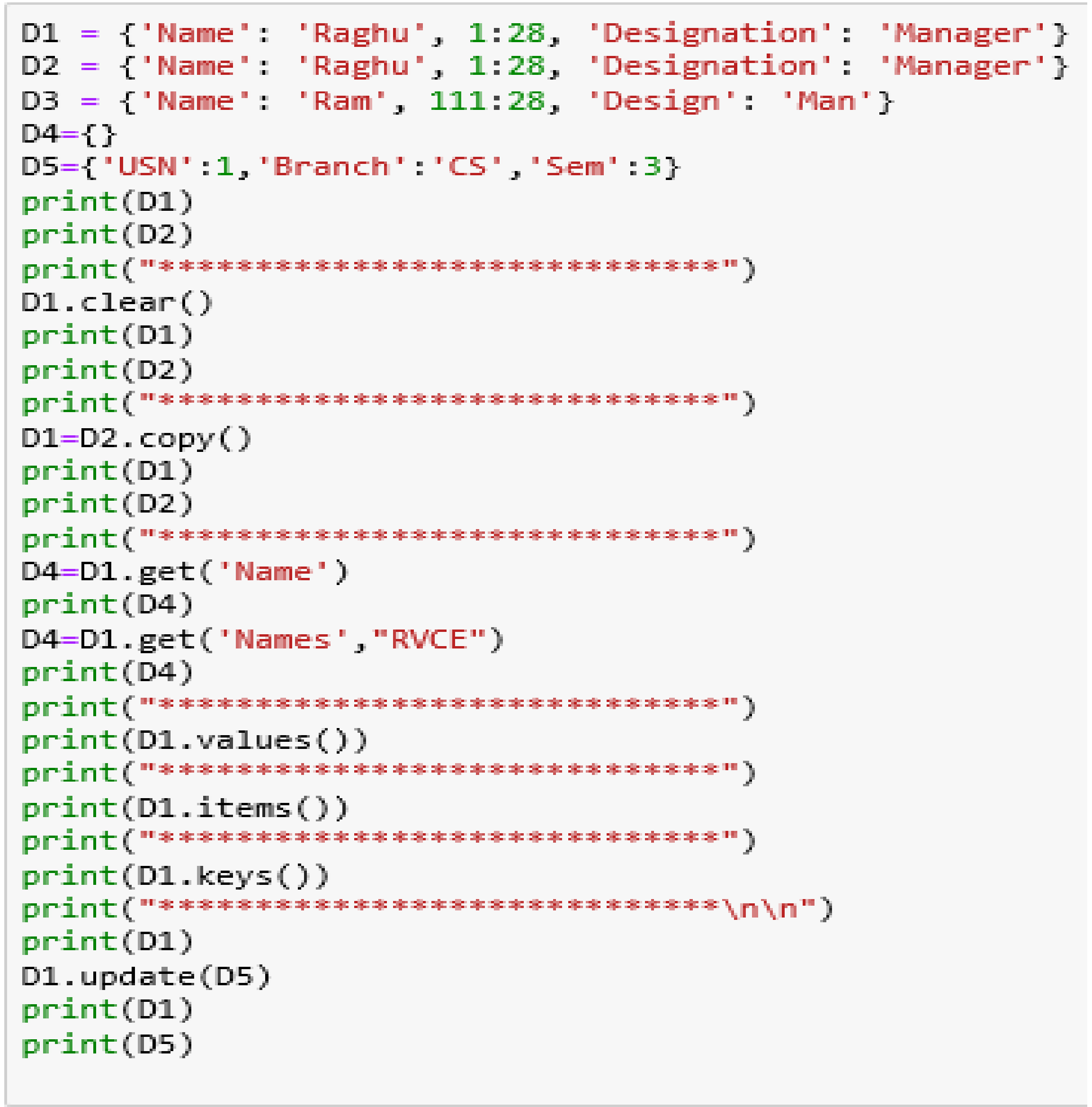
- type(variable) Returns the type of the passed variable.



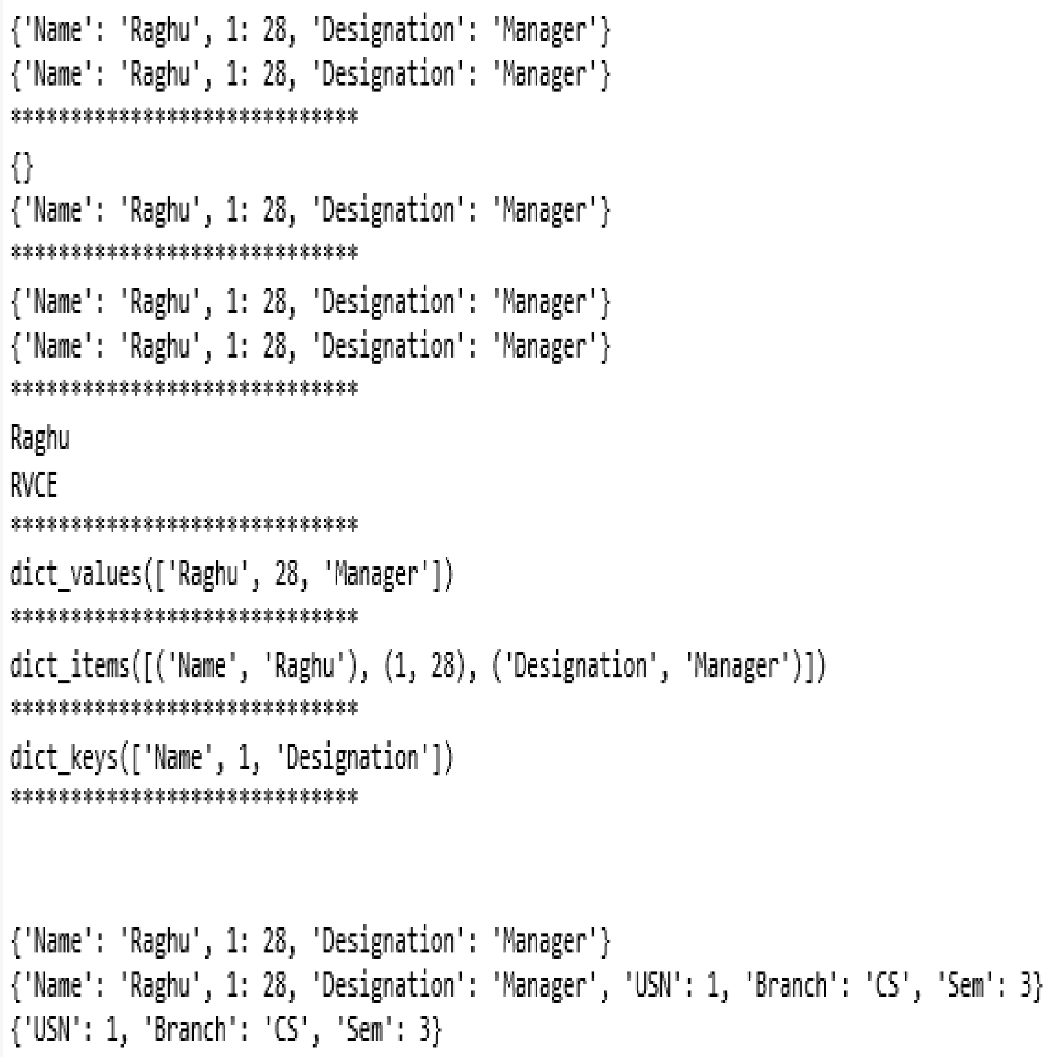
```
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
A= ["a",1,4.6]
B=("a","cv",1)
print(type(D))
print(type(A))
print(type(B))
```

```
<class 'dict'>
<class 'list'>
<class 'tuple'>
```

Built-in dictionary methods



```
D1 = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
D2 = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
D3 = {'Name': 'Ram', 111:28, 'Design': 'Man'}
D4={}
D5={'USN':1,'Branch':'CS','Sem':3}
print(D1)
print(D2)
print("*****************************")
D1.clear()
print(D1)
print(D2)
print("*****************************")
D1=D2.copy()
print(D1)
print(D2)
print("*****************************")
D4=D1.get('Name')
print(D4)
D4=D1.get('Names',"RVCE")
print(D4)
print("*****************************")
print(D1.values())
print("*****************************")
print(D1.items())
print("*****************************")
print(D1.keys())
print("*******************************\n\n")
print(D1)
D1.update(D5)
print(D1)
print(D5)
```



```
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
*****************************
{}
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
*****************************
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
*****************************
Raghu
RVCE
*****************************
dict_values(['Raghu', 28, 'Manager'])
*****************************
dict_items([('Name', 'Raghu'), (1, 28), ('Designation', 'Manager')])
*****************************
dict_keys(['Name', 1, 'Designation'])
*****************************


{'Name': 'Raghu', 1: 28, 'Designation': 'Manager'}
{'Name': 'Raghu', 1: 28, 'Designation': 'Manager', 'USN': 1, 'Branch': 'CS', 'Sem': 3}
{'USN': 1, 'Branch': 'CS', 'Sem': 3}
```

THANK YOU