Python Programming

Parul® University

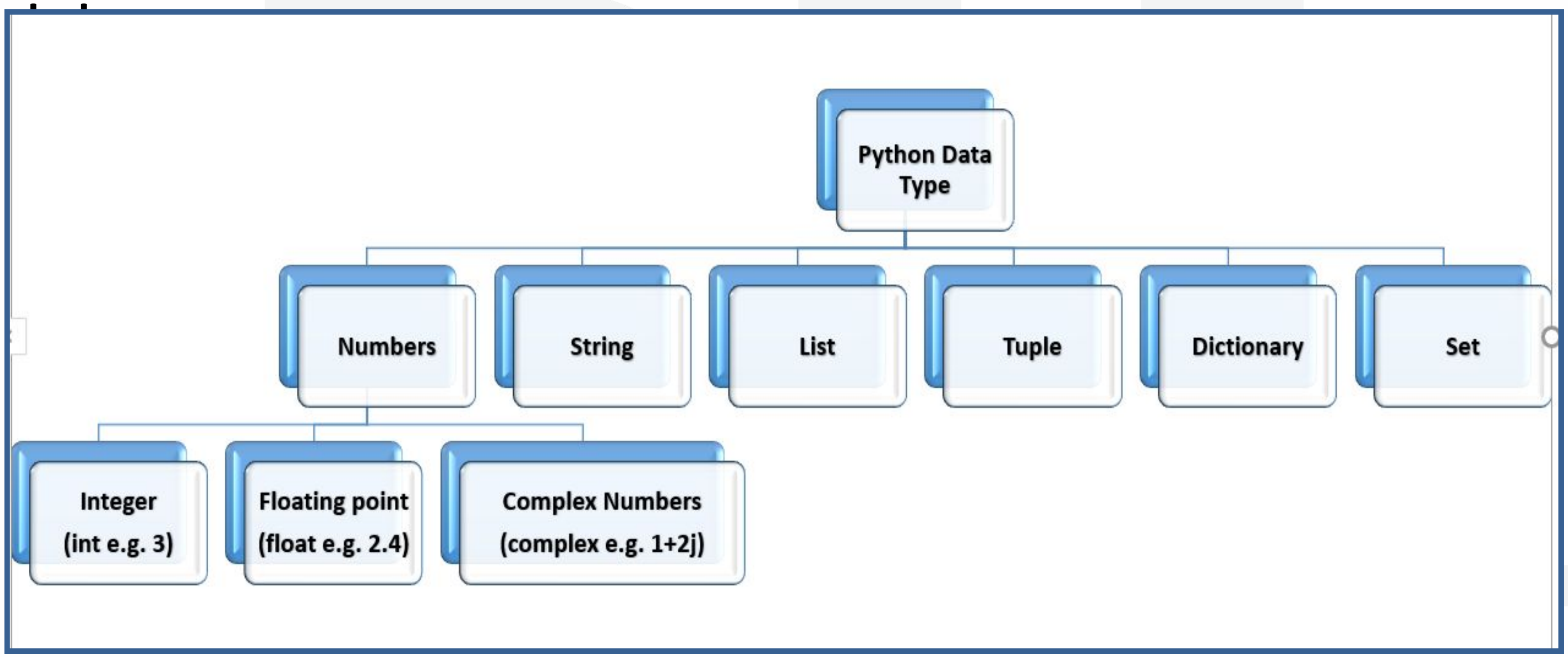
DIGITAL LEARNING CONTENT

CHAPTER-2

Python Data Types

Data Types

- Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

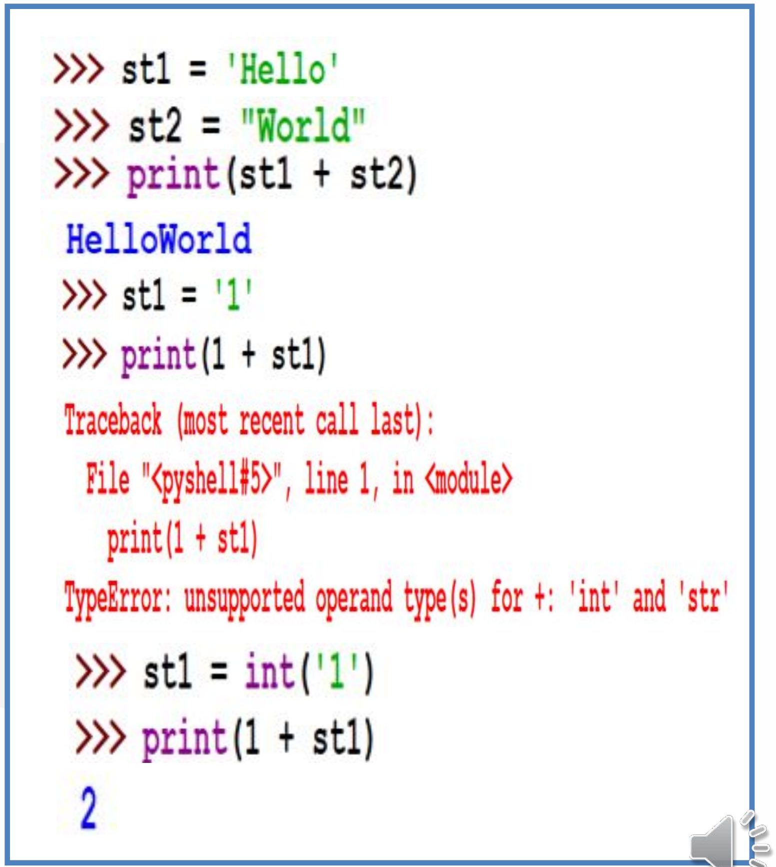


Parul® University

DIGITAL LEARNING CONTENT

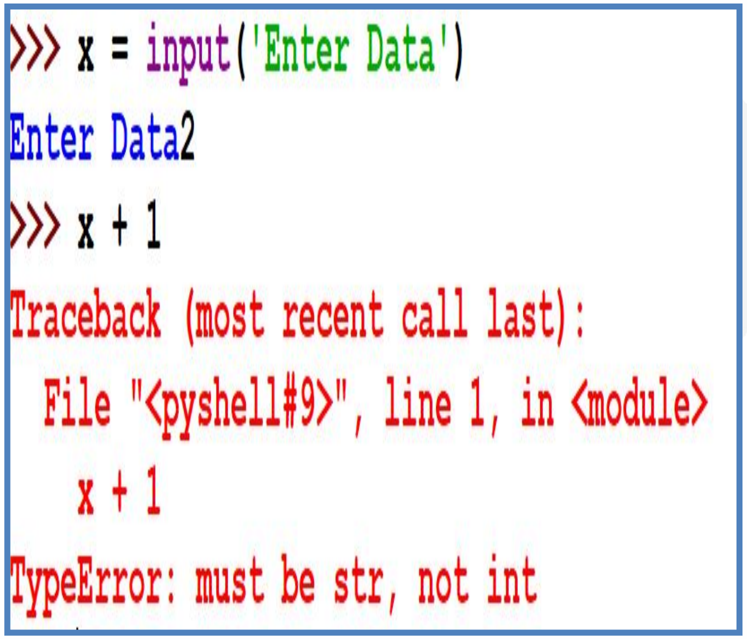
String

- Sequence of characters
- Keyword for string datatype : str
- Uses single or double quotes : 'Hello' or "Hello"
- When a string contains numbers, it is still a string : ‘123’
- Convert numbers in a string into a number using int()
- immutable : Value of sting can not be changed

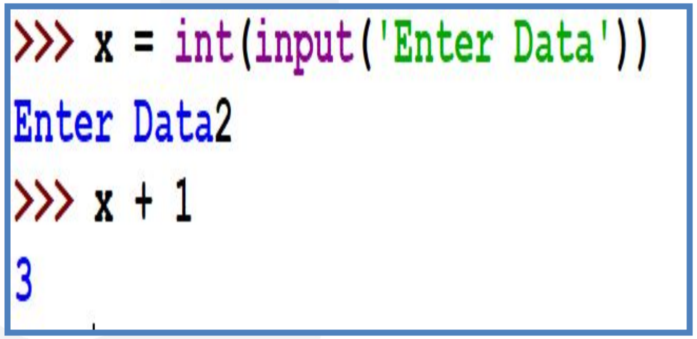


```
>>> st1 = 'Hello'
>>> st2 = "World"
>>> print(st1 + st2)
HelloWorld
>>> st1 = '1'
>>> print(1 + st1)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print(1 + st1)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> st1 = int('1')
>>> print(1 + st1)
2
```

input() reads string only



```
>>> x = input('Enter Data')
Enter Data2
>>> x + 1
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    x + 1
TypeError: must be str, not int
```



```
>>> x = int(input('Enter Data'))
Enter Data2
>>> x + 1
3
```

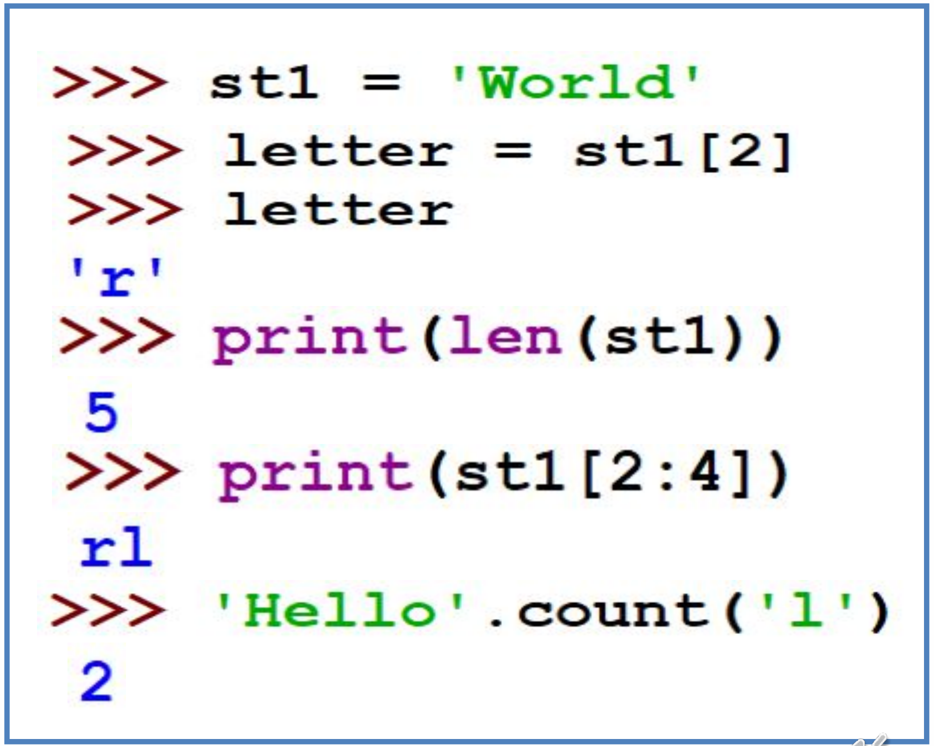
Parul® University

DIGITAL LEARNING CONTENT

More on String

- Get length of string
  - len()
- String slicing using colon operator
  - st[start : end]
- Count the occurrence of character
  - st.count(‘character)
- String uses index
  - String = ‘World’ is indexed as follows

W	o	r	l	d
0	1	2	3	4



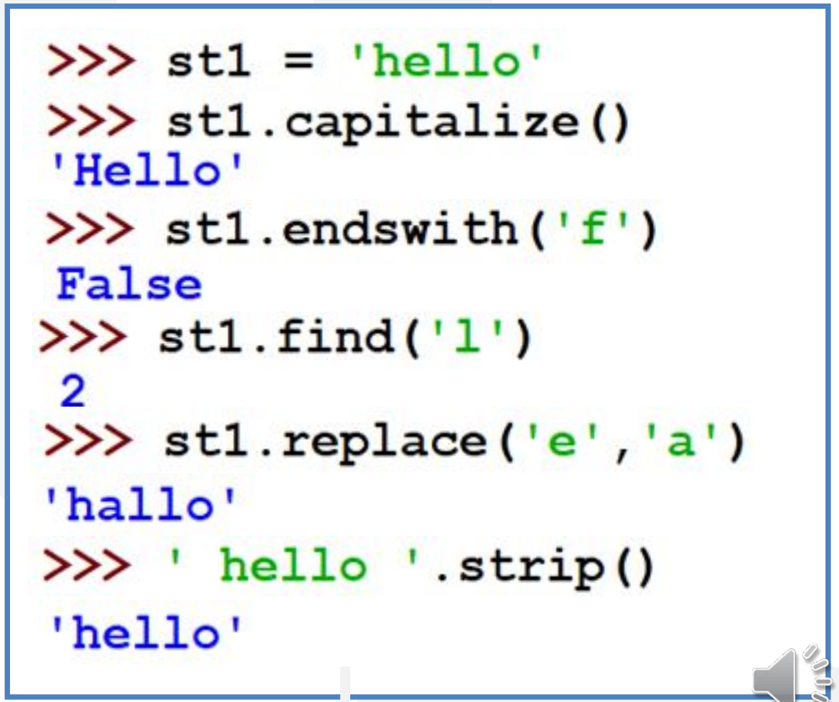
```
>>> st1 = 'World'
>>> letter = st1[2]
>>> letter
'r'
>>> print(len(st1))
5
>>> print(st1[2:4])
rl
>>> 'Hello'.count('l')
2
```

String Library

- List of common function provided by string library
- Explore more using : dir(string_object)

- capitalize
- center
- count
- endswith
- find
- Index
- Isalnum
- Isalpha
- Isdigit
- Islower
- isupper

- join
- ljust
- Lower
- Lstrip
- replace
- rjust
- rsplit
- Rstrip
- Startswith
- Swapcase
- upper



```
>>> st1 = 'hello'
>>> st1.capitalize()
'Hello'
>>> st1.endswith('f')
False
>>> st1.find('l')
2
>>> st1.replace('e','a')
'hallo'
>>> ' hello '.strip()
'hello'
```

Parul® University

DIGITAL LEARNING CONTENT

List

- Collection of many values in a single variable
- Mutable : value of list variable can be changed
- Uses square brackets : []
- Example:

list_of_number = [1, 2.3, 3, 4, 0]

Friend_list = [‘kanu’, ‘manu’, ‘tanu’]

City_list = [‘Baroda’, ‘Anand’, 123]

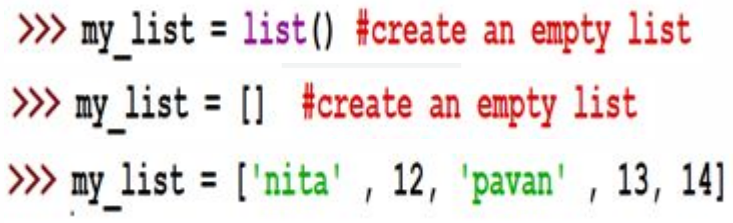
Image source : Google

Parul® University

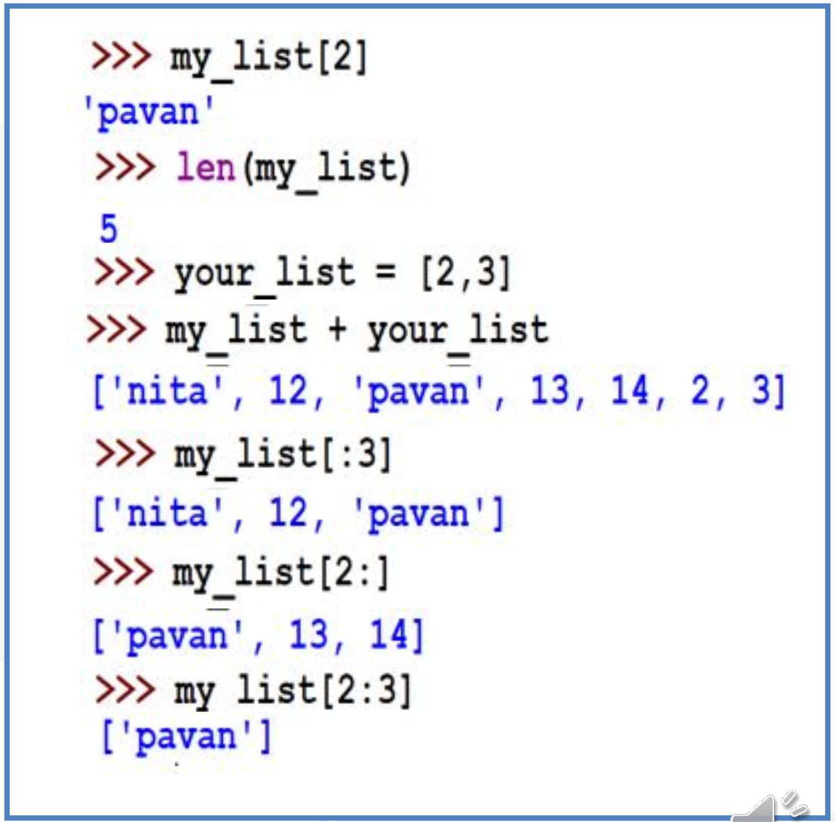
Exploring List

- List is an ordered collection
- Access any element using index
- Get number of elements : len()
- List concatenation : ‘+’
- List slicing using colon

nita	12	pavan	13	14
0	1	2	3	4



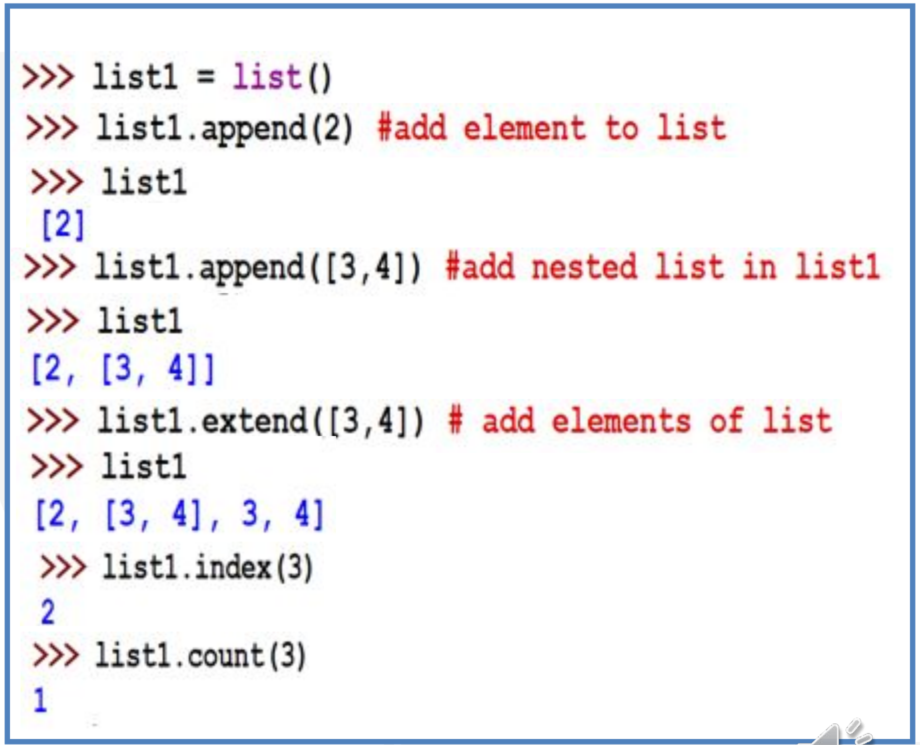
```
>>> my_list = list() #create an empty list
>>> my_list = []  #create an empty list
>>> my_list = ['nita' , 12, 'pavan' , 13, 14]
```



```
>>> my_list[2]
'pavan'
>>> len(my_list)
5
>>> your_list = [2,3]
>>> my_list + your_list
['nita', 12, 'pavan', 13, 14, 2, 3]
>>> my_list[:3]
['nita', 12, 'pavan']
>>> my_list[2:]
['pavan', 13, 14]
>>> my list[2:3]
['pavan']
```

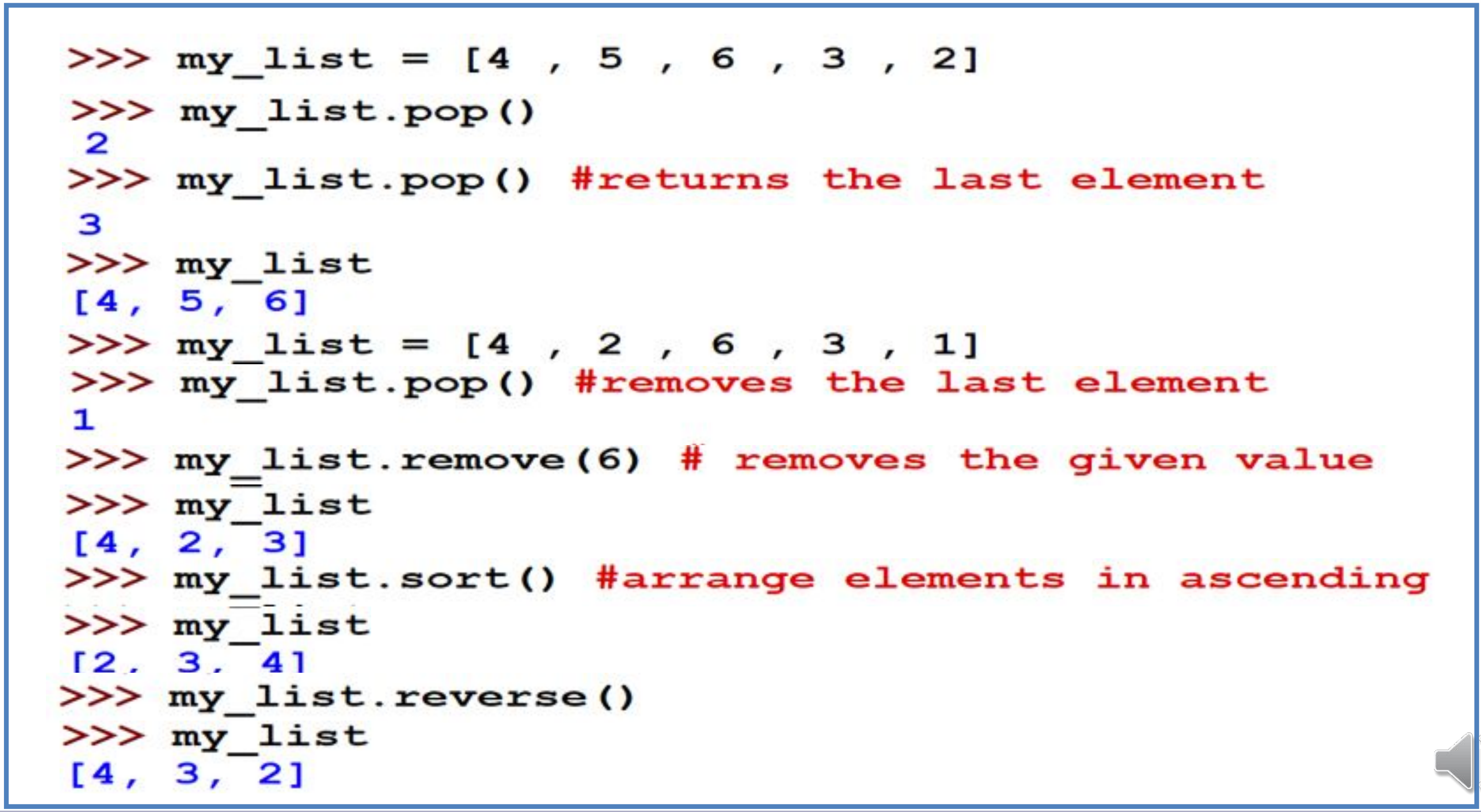
List Methods

Methods	Description
append	Add element in the list as it is
count	Count the occurrence of an element in list
extend	Add values of object as elements of the list
index	Get the index of an element
sort	Sort the elements of list
insert	Add element at specified index
pop	Retrieve the last element of list
remove	Remove the given element from list
reverse	Reverse the sequence of elements in list
clear	Empty the list by removing all elements



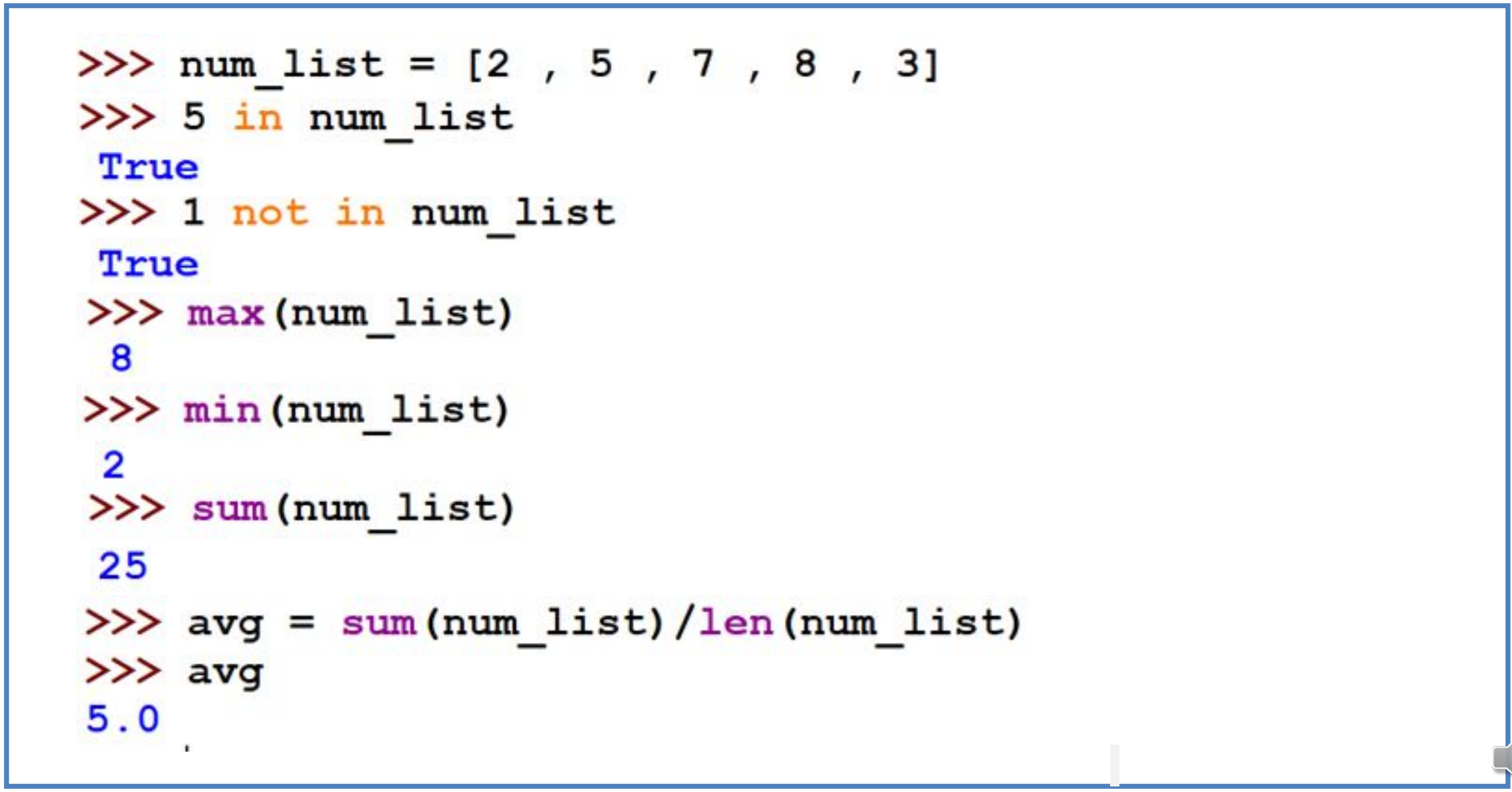
```
>>> list1 = list()
>>> list1.append(2) #add element to list
>>> list1
[2]
>>> list1.append([3,4]) #add nested list in list1
>>> list1
[2, [3, 4]]
>>> list1.extend([3,4]) # add elements of list
>>> list1
[2, [3, 4], 3, 4]
>>> list1.index(3)
2
>>> list1.count(3)
1
```

Playing with List



```
>>> my_list = [4 , 5 , 6 , 3 , 2]
>>> my_list.pop()
 2
>>> my_list.pop() #returns the last element
 3
>>> my_list
[4, 5, 6]
>>> my_list = [4 , 2 , 6 , 3 , 1]
>>> my_list.pop() #removes the last element
1
>>> my_list.remove(6) # removes the given value
>>> my_list
[4, 2, 3]
>>> my_list.sort() #arrange elements in ascending
>>> my_list
[2, 3, 4]
>>> my_list.reverse()
>>> my_list
[4, 3, 2]
```

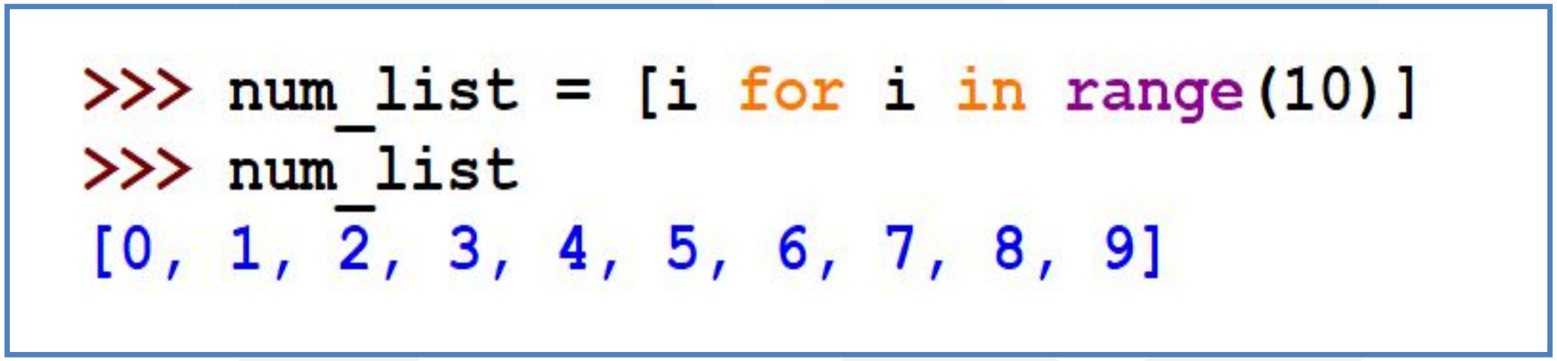
Playing with List



```
>>> num_list = [2 , 5 , 7 , 8 , 3]
>>> 5 in num_list
 True
>>> 1 not in num_list
 True
 >>> max(num_list)
  8
>>> min(num_list)
 2
 >>> sum(num_list)
 25
 >>> avg = sum(num_list)/len(num_list)
 >>> avg
 5.0
```

List Comprehension

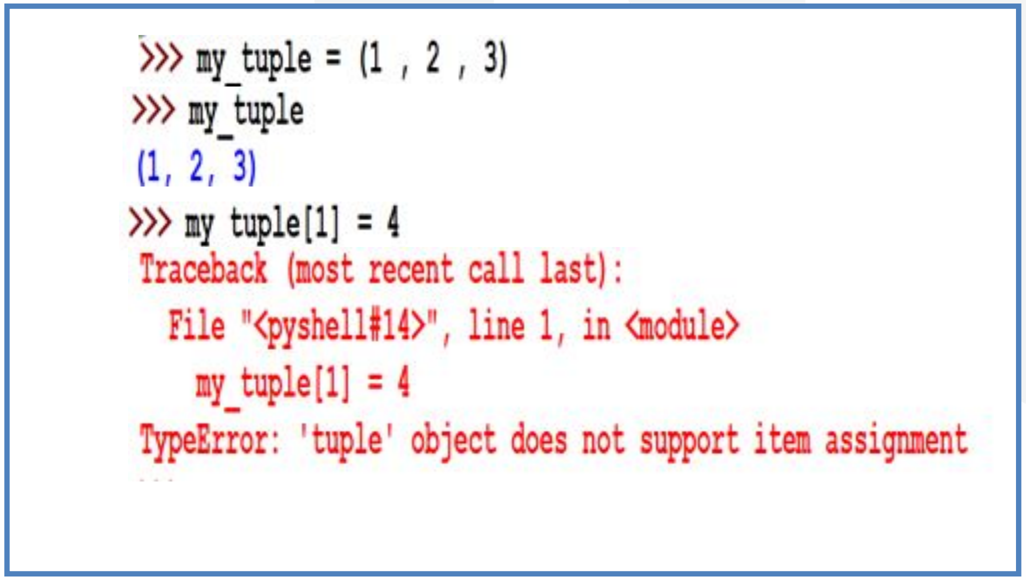
- Use logical statement to create list



```
>>> num_list = [i for i in range(10)]
>>> num_list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Tuple

- Same as list
- Immutable : values can not be changed
- Use round brackets



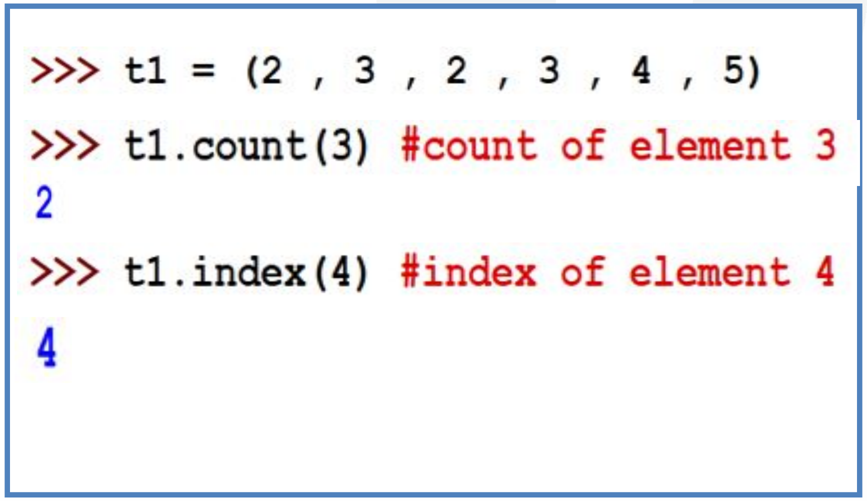
```
>>> my_tuple = (1 , 2 , 3)
>>> my_tuple
(1, 2, 3)
>>> my tuple[1] = 4
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    my_tuple[1] = 4
TypeError: 'tuple' object does not support item assignment
```

Parul® University

DIGITAL LEARNING CONTENT

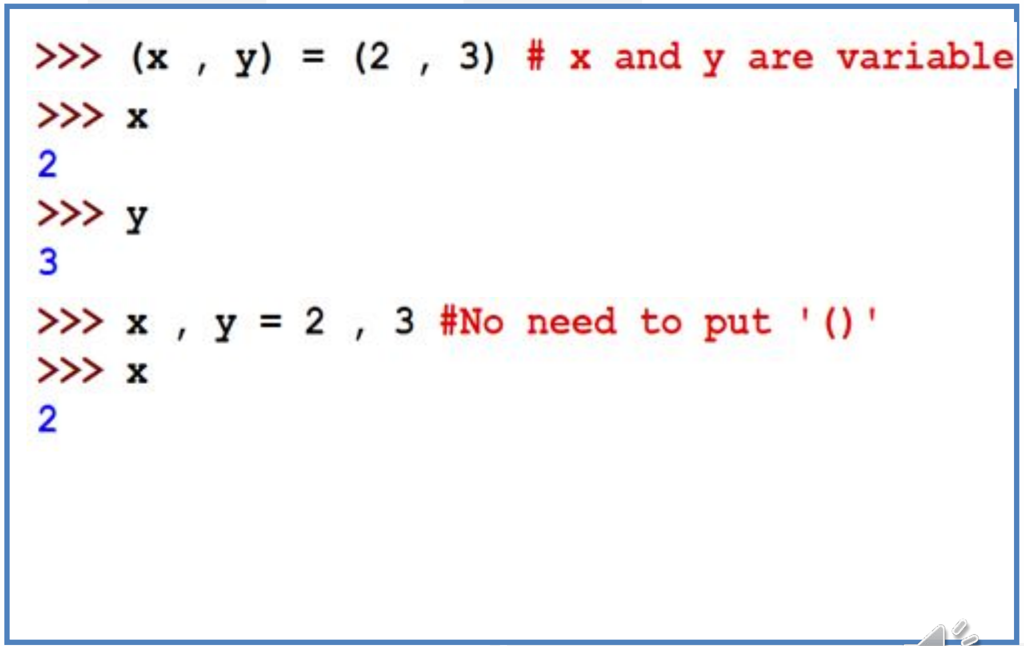
Exploring Tuple

- Tuple has two method
   - Count
   - Index



```
>>> t1 = (2 , 3 , 2 , 3 , 4 , 5)
>>> t1.count(3) #count of element 3
2
>>> t1.index(4) #index of element 4
4
```

- Tuple as assignment



```
>>> (x , y) = (2 , 3) # x and y are variable
>>> x
2
>>> y
3
>>> x , y = 2 , 3 #No need to put '()'
>>> x
2
```

Dictionary

- Unordered collection of data in key : value form
- Indexed by unique key
- Uses curly braces : { }

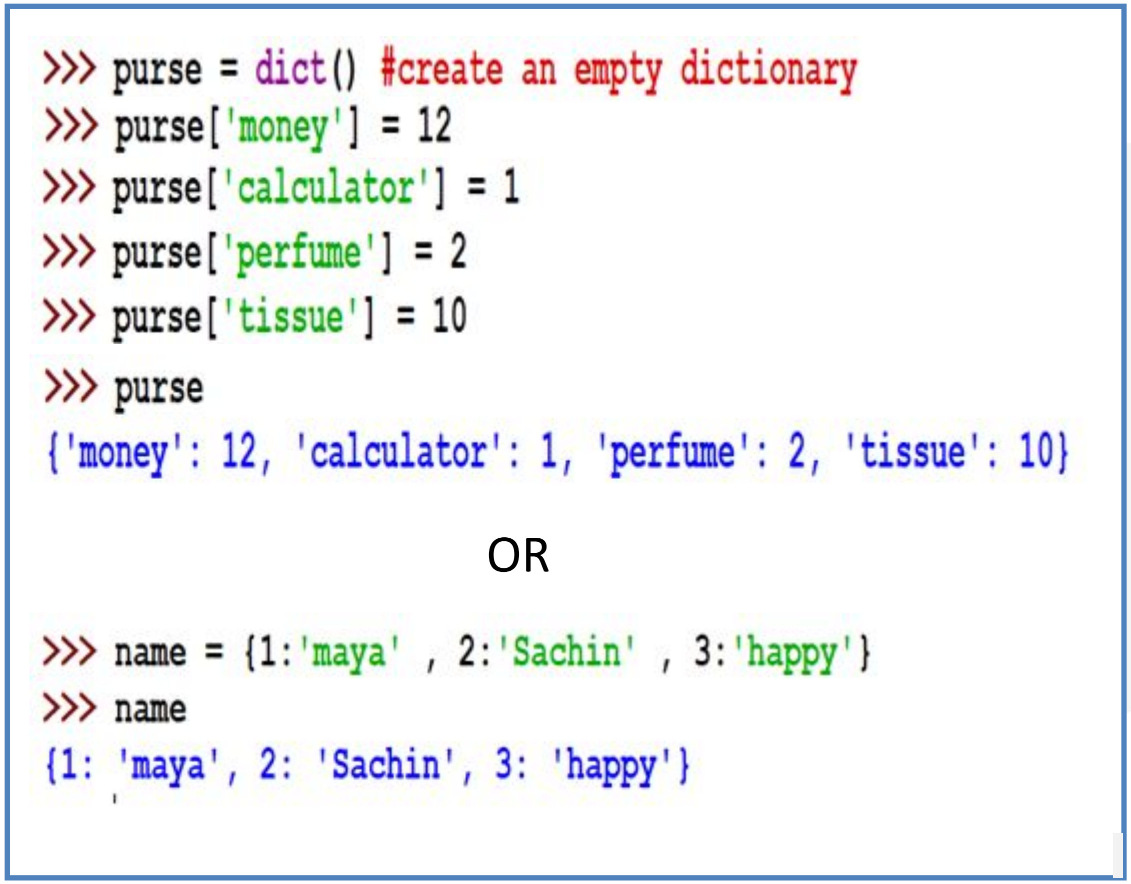
Bag of items

Image source : Google

Parul® University

DIGITAL LEARNING CONTENT

Creating Dictionary

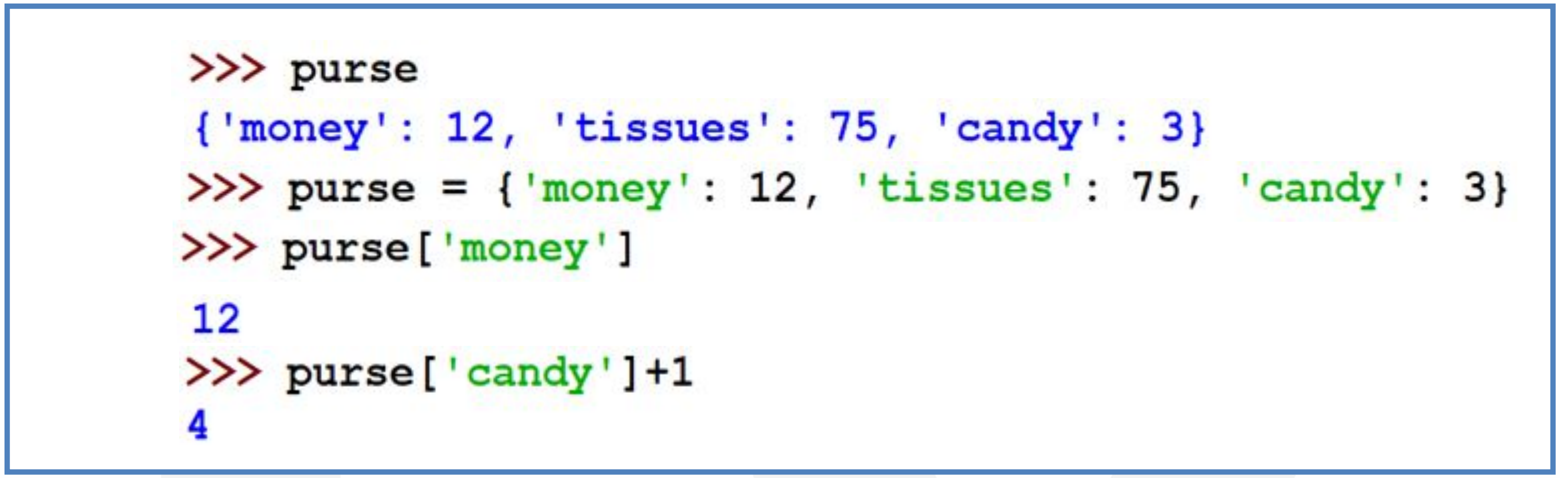


```
>>> purse = dict() #create an empty dictionary
>>> purse['money'] = 12
>>> purse['calculator'] = 1
>>> purse['perfume'] = 2
>>> purse['tissue'] = 10
>>> purse
{'money': 12, 'calculator': 1, 'perfume': 2, 'tissue': 10}
```

OR

```
>>> name = {1:'maya' , 2:'Sachin' , 3:'happy'}
>>> name
{1: 'maya', 2: 'Sachin', 3: 'happy'}
```

Keys as Index



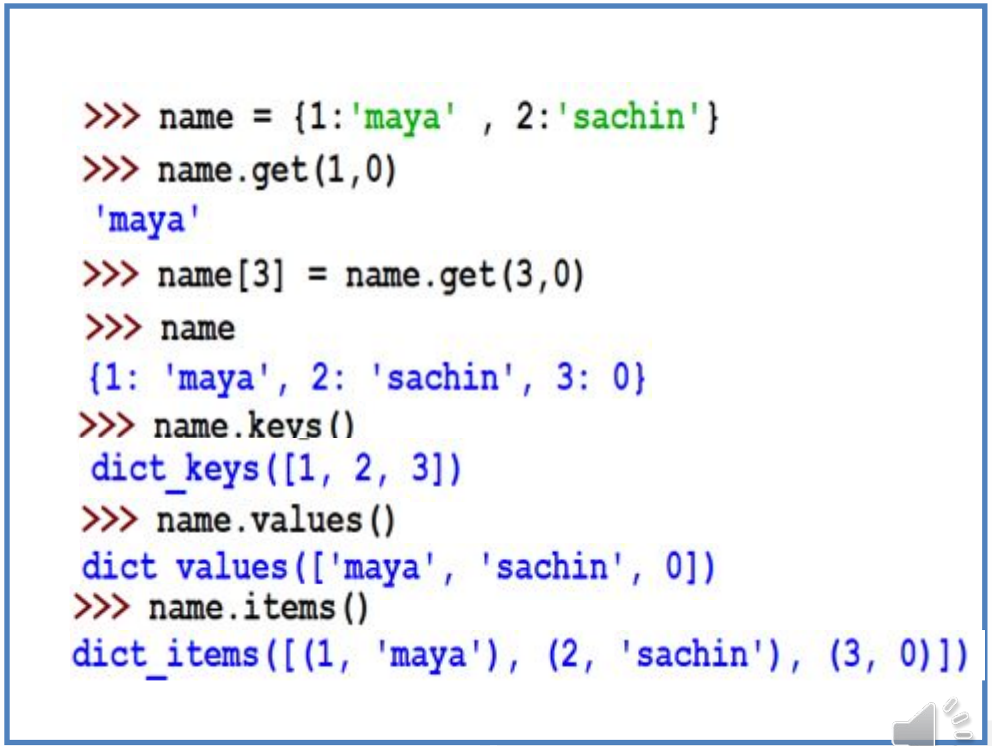
```
>>> purse
{'money': 12, 'tissues': 75, 'candy': 3}
>>> purse = {'money': 12, 'tissues': 75, 'candy': 3}
>>> purse['money']
12
>>> purse['candy']+1
4
```

12	75	3
money	tissues	candy

Dictionary Methods

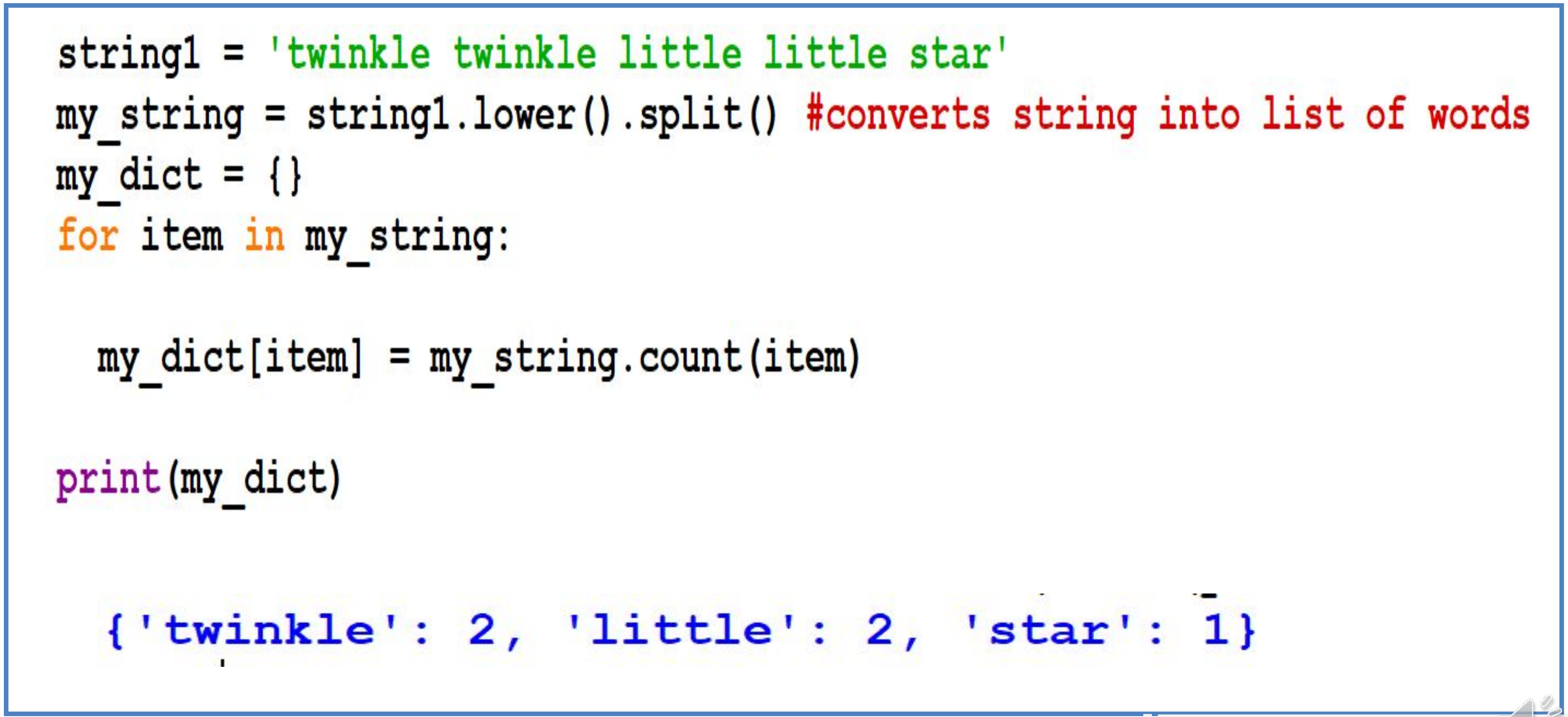
- **get()**
  - give the value at given key if key is there, otherwise create given and assign default value

- **keys()** : list of keys
- **values()** : list of values
- **Items()** : list of (key, value)



```
>>> name = {1:'maya' , 2:'sachin'}
>>> name.get(1,0)
'maya'
>>> name[3] = name.get(3,0)
>>> name
{1: 'maya', 2: 'sachin', 3: 0}
>>> name.keys()
dict_keys([1, 2, 3])
>>> name.values()
dict_values(['maya', 'sachin', 0])
>>> name.items()
dict_items([(1, 'maya'), (2, 'sachin'), (3, 0)])
```

Counting Pattern



```python
string1 = 'twinkle twinkle little little star'
my_string = string1.lower().split() #converts string into list of words
my_dict = {}
for item in my_string:

  my_dict[item] = my_string.count(item)

print(my_dict)
```

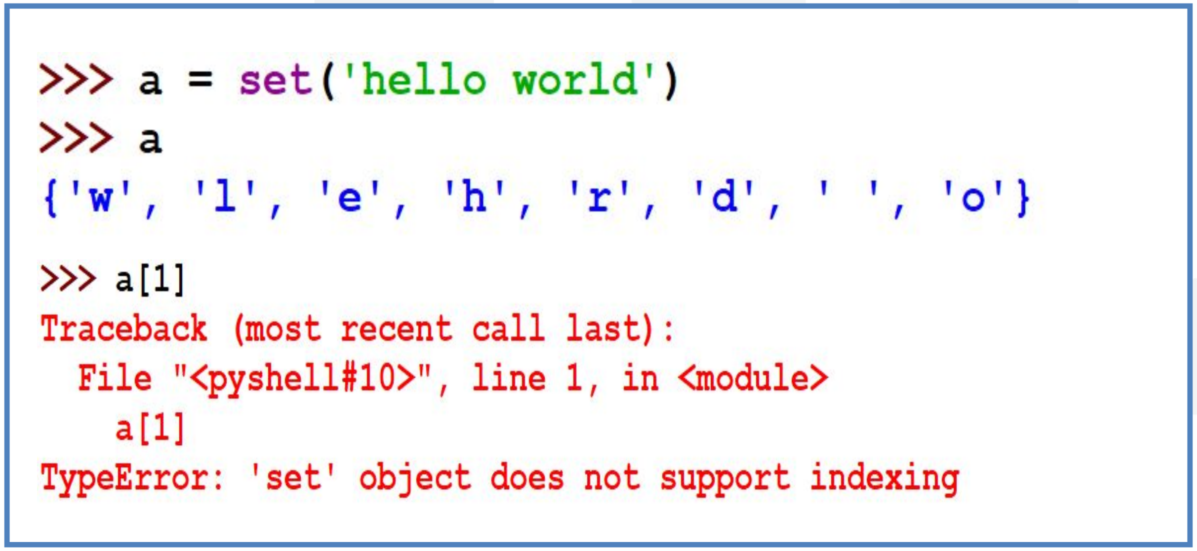
```
{'twinkle': 2, 'little': 2, 'star': 1}
```

Parul® University

DIGITAL LEARNING CONTENT

Set

- Unordered collection of unique and immutable objects
- set itself is mutable
- Uses curly braces : {}



```
>>> a = set('hello world')
>>> a
{'w', 'l', 'e', 'h', 'r', 'd', ' ', 'o'}

>>> a[1]
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    a[1]
TypeError: 'set' object does not support indexing
```

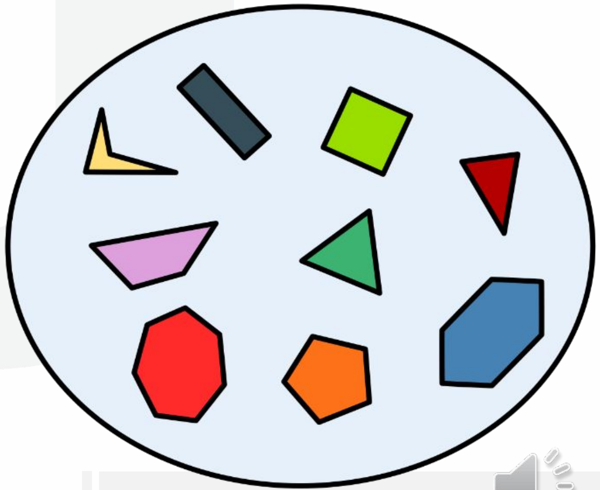
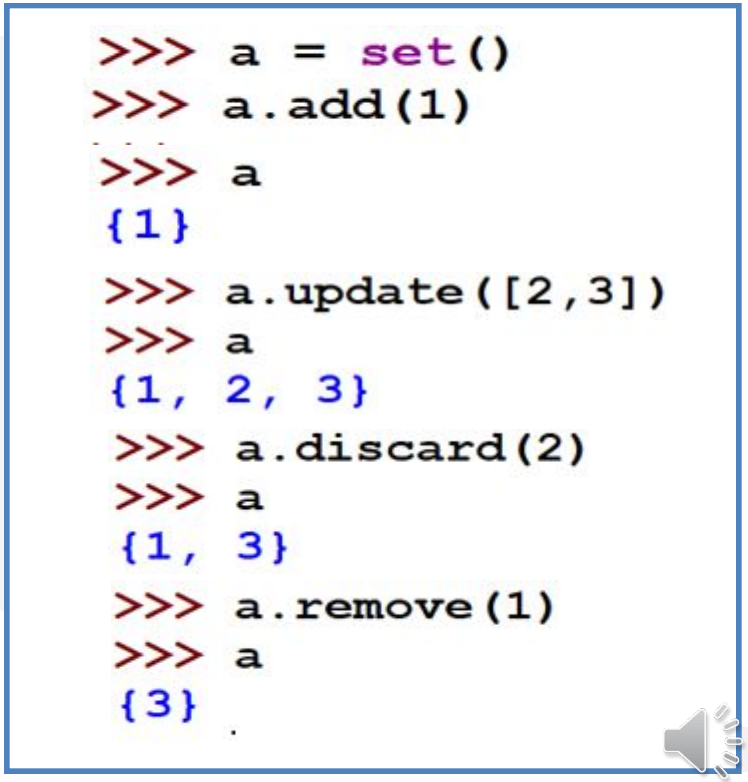


Image source : Google

Exploring Set

- **add()** : add any single element in set
- **update()** : add multiple elements passed in the form of tuples, list, string or other set in set
- **discard()/ remove()** : remove element from set



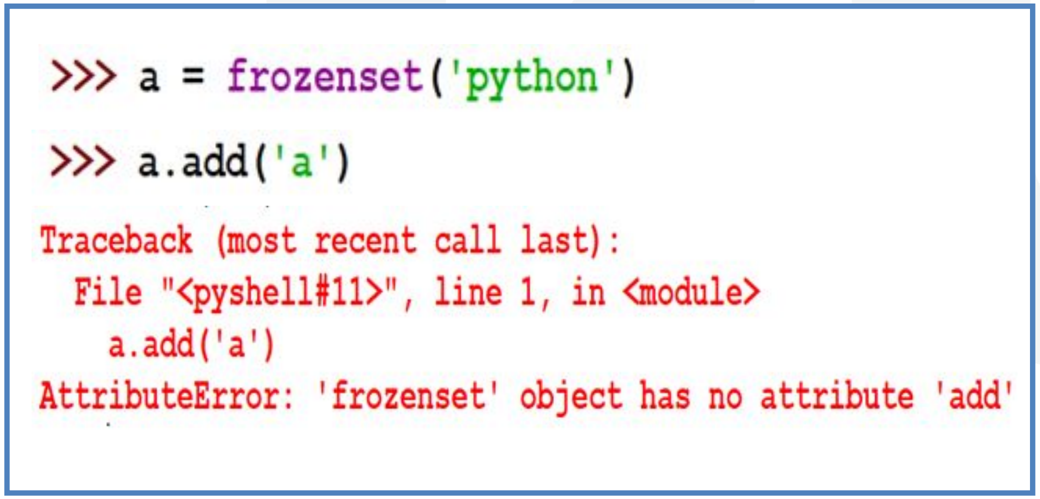
```
>>> a = set()
>>> a.add(1)
>>> a
{1}
>>> a.update([2,3])
>>> a
{1, 2, 3}
>>> a.discard(2)
>>> a
{1, 3}
>>> a.remove(1)
>>> a
{3}
```

Parul® University

DIGITAL LEARNING CONTENT

Frozenset

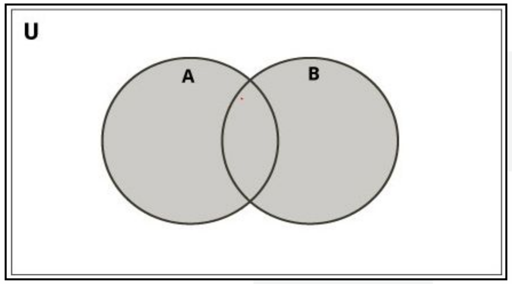
- Frozensets are like sets except that they cannot be changed
- They are immutable



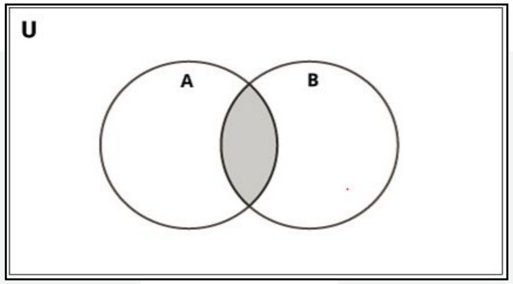
```
>>> a = frozenset('python')
>>> a.add('a')
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    a.add('a')
AttributeError: 'frozenset' object has no attribute 'add'
```

Image source : Google

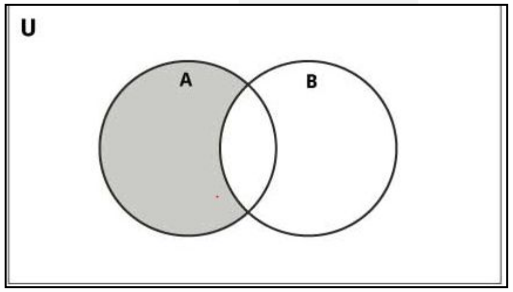
Set Operation



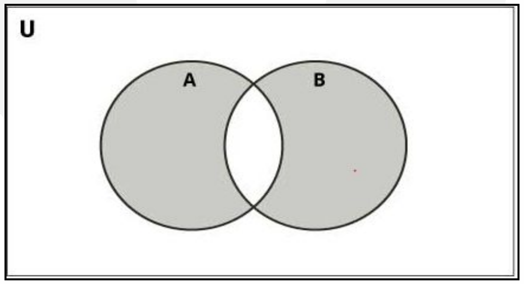
union (|)



intersection (&)



difference (-)



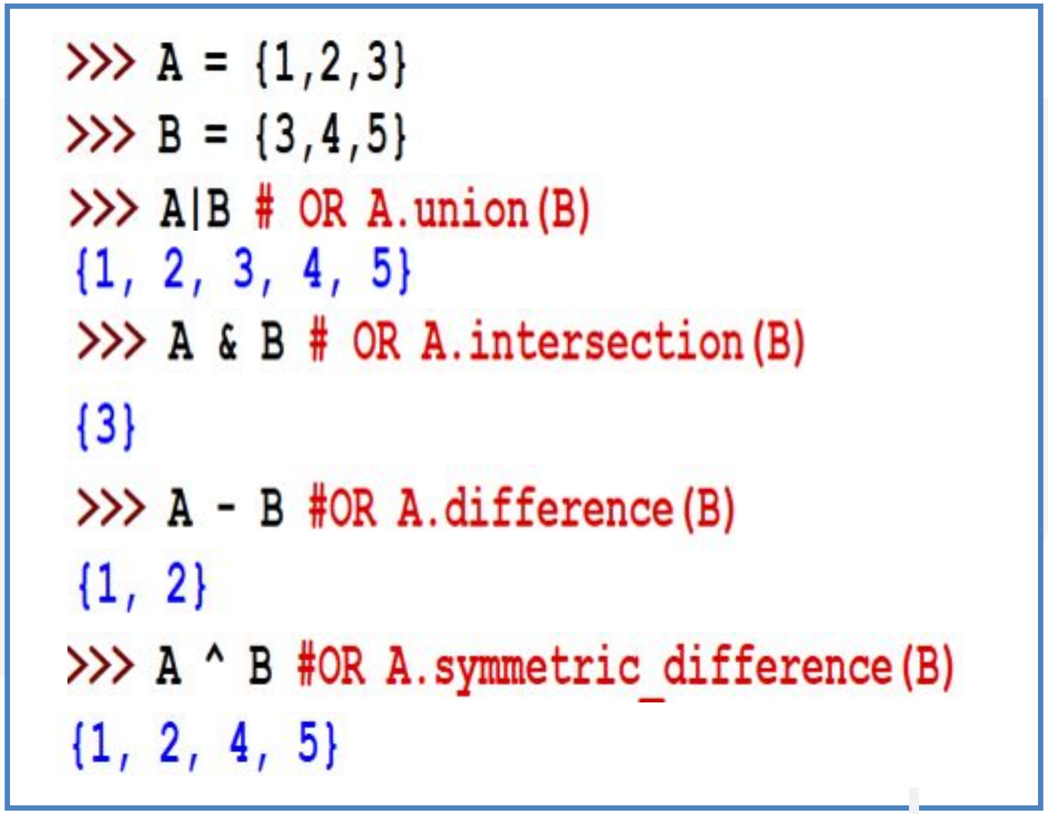
symmetric_difference (^)

Image source : Google

Parul® University

DIGITAL LEARNING CONTENT

Set Operation



```
>>> A = {1,2,3}
>>> B = {3,4,5}
>>> A|B # OR A.union(B)
{1, 2, 3, 4, 5}
>>> A & B # OR A.intersection(B)
{3}
>>> A - B #OR A.difference(B)
{1, 2}
>>> A ^ B #OR A.symmetric_difference(B)
{1, 2, 4, 5}
```

Operators

- Required to perform an operation on variables

Arithmetic Operators

Operators	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Reminder
//	Integer division

Table 1.1 Arithmetic Operators

Comparison Operators

Operators	Name
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Table 1.2 Comparison Operators

Parul® University

DIGITAL LEARNING CONTENT

Operators

Logical Operators

Operator	Operation
and	(x and y) is true if both x and y are true
or	(x or y) is true if either x or y is true
not	True value will become false

Table 1.3 Logical Operators

Special Operators

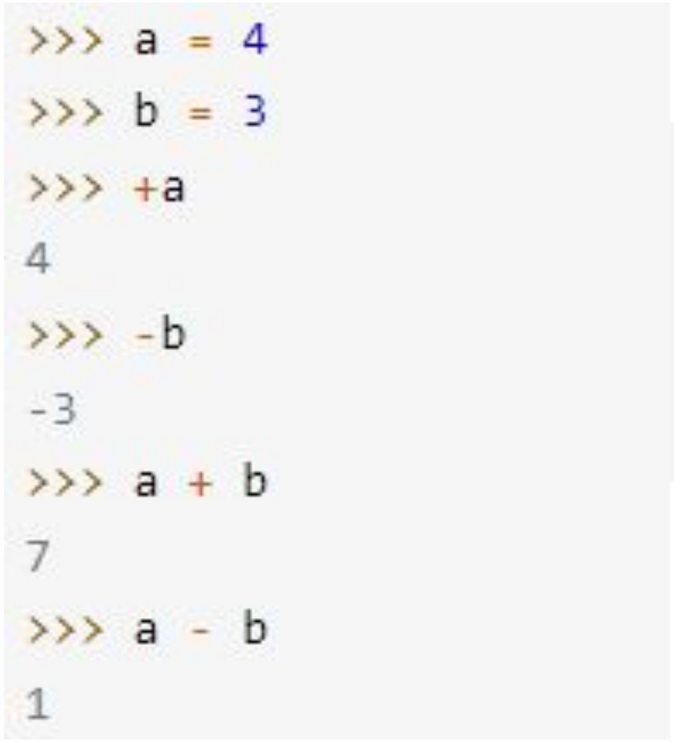
Operator	Operation
is	Returns true if identity of two operands are same, else false
is not	Returns true if identity of two operands are not same, else false
in	Returns True if a sequence with the specified value is present in the object
not in	Returns True if a sequence with the specified value is not present in the object

Table 1.4 Special Operators

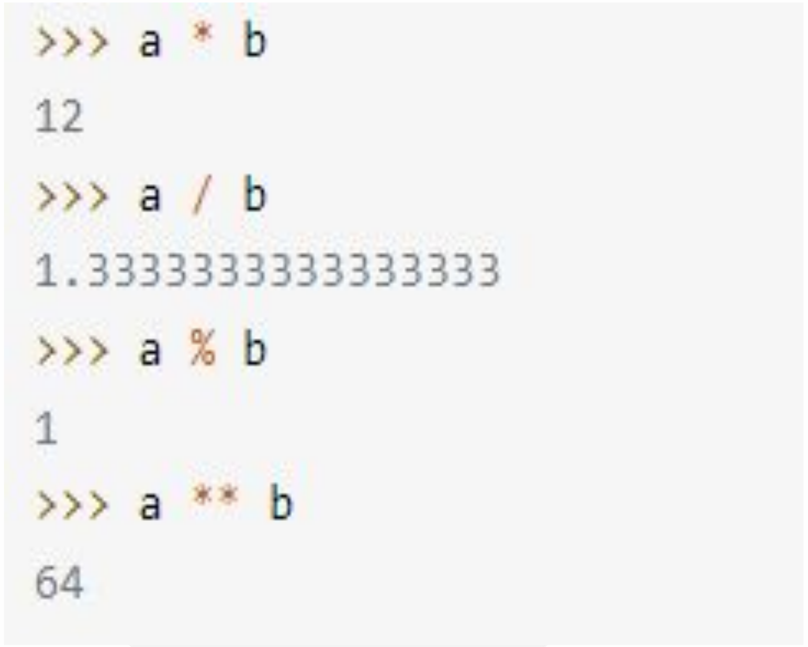
Parul® University

DIGITAL LEARNING CONTENT

Python Expression

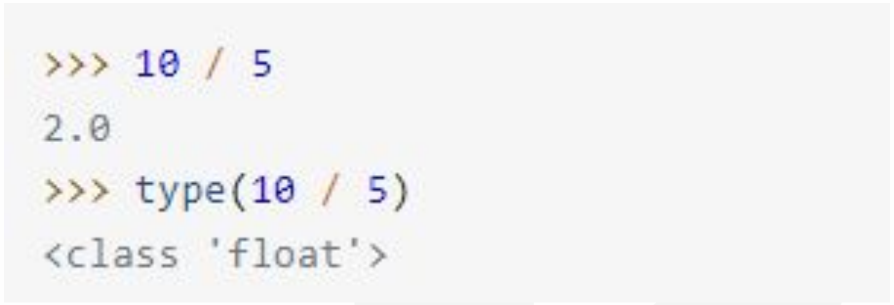


```
>>> a = 4
>>> b = 3
>>> +a
4
>>> -b
-3
>>> a + b
7
>>> a - b
1
```

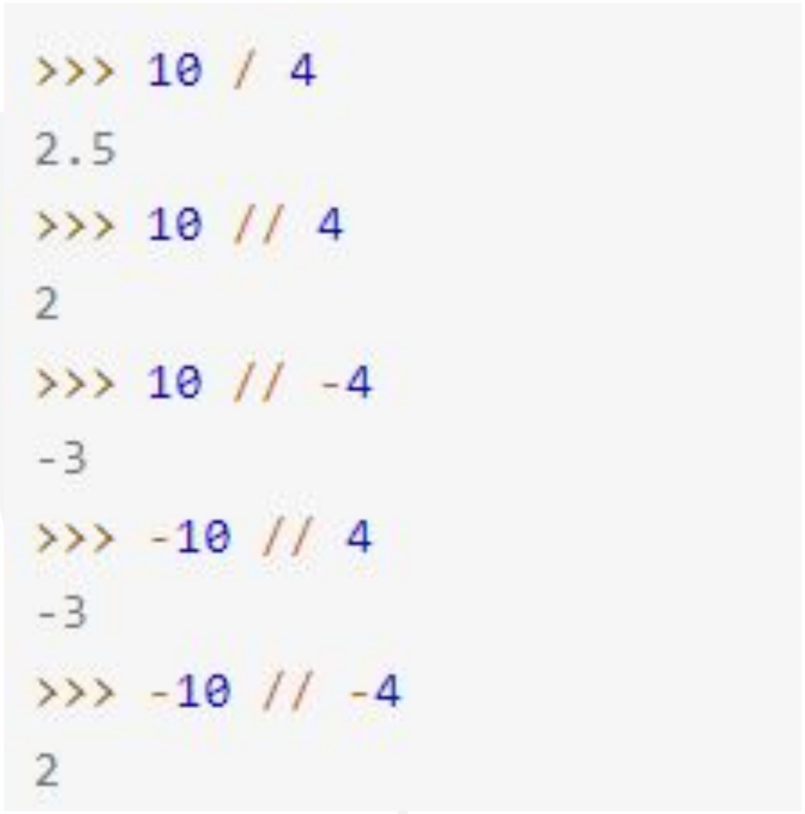


```
>>> a * b
12
>>> a / b
1.3333333333333333
>>> a % b
1
>>> a ** b
64
```

Python Expression

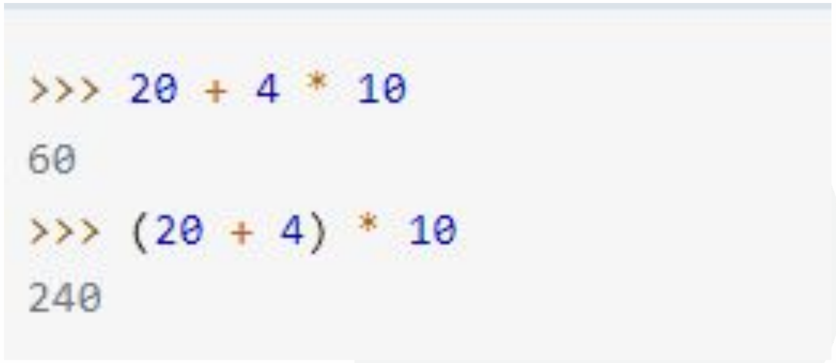


```
>>> 10 / 5
2.0
>>> type(10 / 5)
<class 'float'>
```

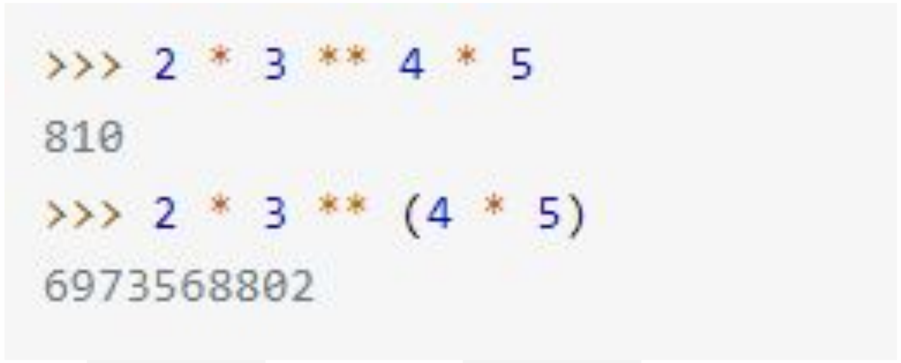


```
>>> 10 / 4
2.5
>>> 10 // 4
2
>>> 10 // -4
-3
>>> -10 // 4
-3
>>> -10 // -4
2
```

Python Expression



```
>>> 20 + 4 * 10
60
>>> (20 + 4) * 10
240
```



```
>>> 2 * 3 ** 4 * 5
810
>>> 2 * 3 ** (4 * 5)
6973568802
```

Augmented Assignment		Standard Assignment
`a += 5`	is equivalent to	`a = a + 5`
`a /= 10`	is equivalent to	`a = a / 10`

DIGITAL LEARNING CONTENT