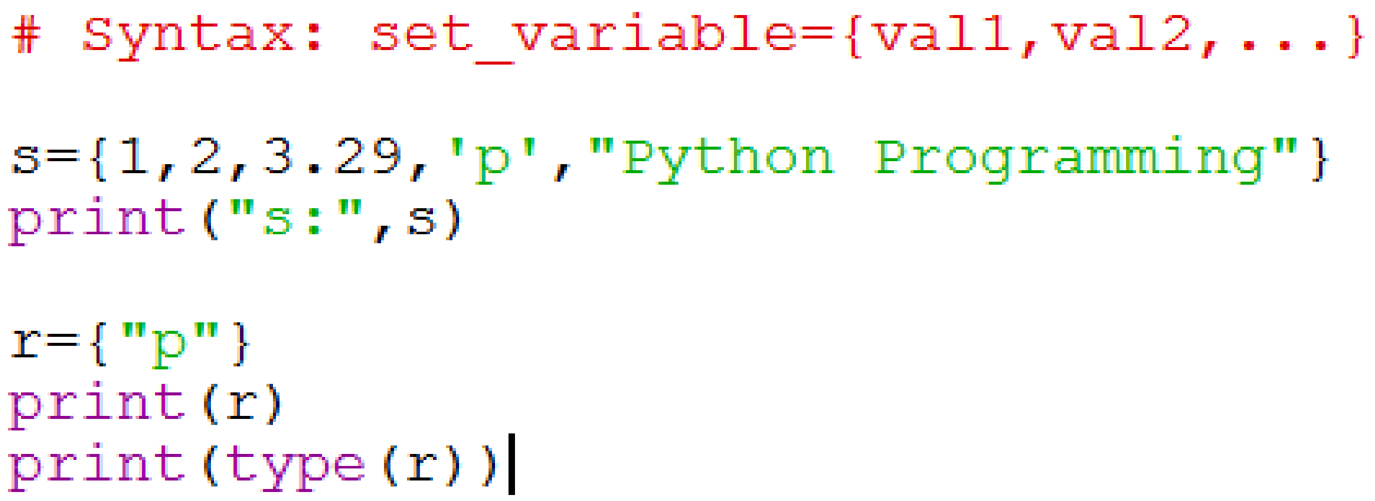
Topics:

- **UNIT – II [14 Lectures]**
- **Data Structures:** Sequence, Lists, Tuple, Sets, Dictionaries
- **Strings and its operations:** Concatenating, Appending, Multiplying strings, Built-in String methods and functions, Slice Operation, Iterating String, String Module
- **Modules:** Importing module, The from..import statement, Name of Module, Making your own modules, The dir()function, The Python Module,Math module, OS Module, Sys Module, Random module
- **Introduction to Functions:** Declaration and Definition, Variable Scope and Lifetime, Return Statements, Types of Arguments, Lambda function, Recursion
- **Functional Programming:** filter() function, map()function, reduce()function

Sets

- A Set is Mutable and un-ordered collection of unique elements
- Sets are lists with no index value and no duplicate entries
- Can be used to identify unique words used in a paragraph
- Operations like intersection, difference, union etc can be performed on sets
- A Set is created by placing all the elements inside curly brackets { },separated by comma or by using built-in function set ( )



```
# Syntax: set_variable={val1,val2,...}

s={1,2,3.29,'p',"Python Programming"}
print("s:",s)

r={"p"}
print(r)
print(type(r))
```

Note: A set can have any number of items and they may be of different data types

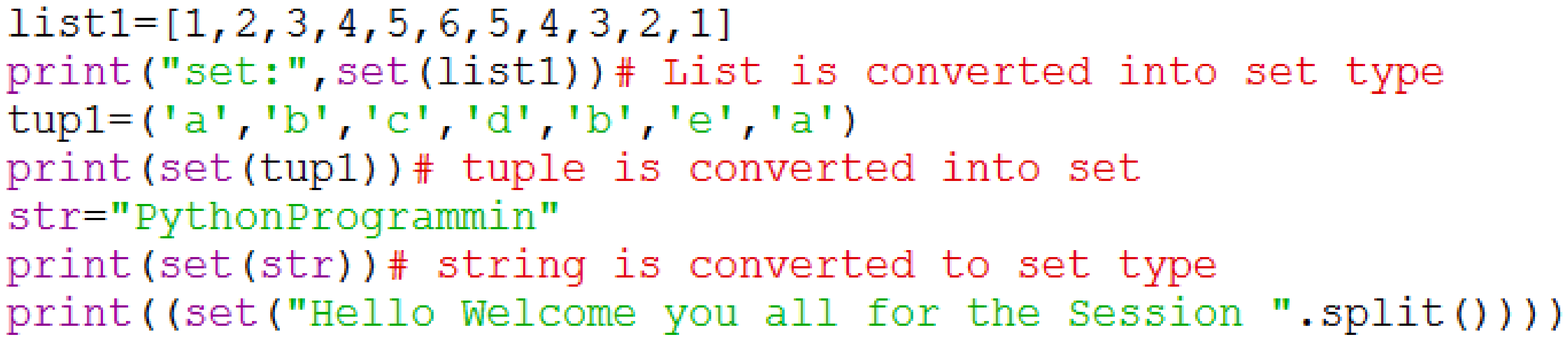
- # program to convert a list of values into a set

```
s=set([1,2,'a','abc',8.72])
print("set elements are:",s)
```

Note: If we add the same element multiple times in as set, they are removed because a set cannot have duplicate values

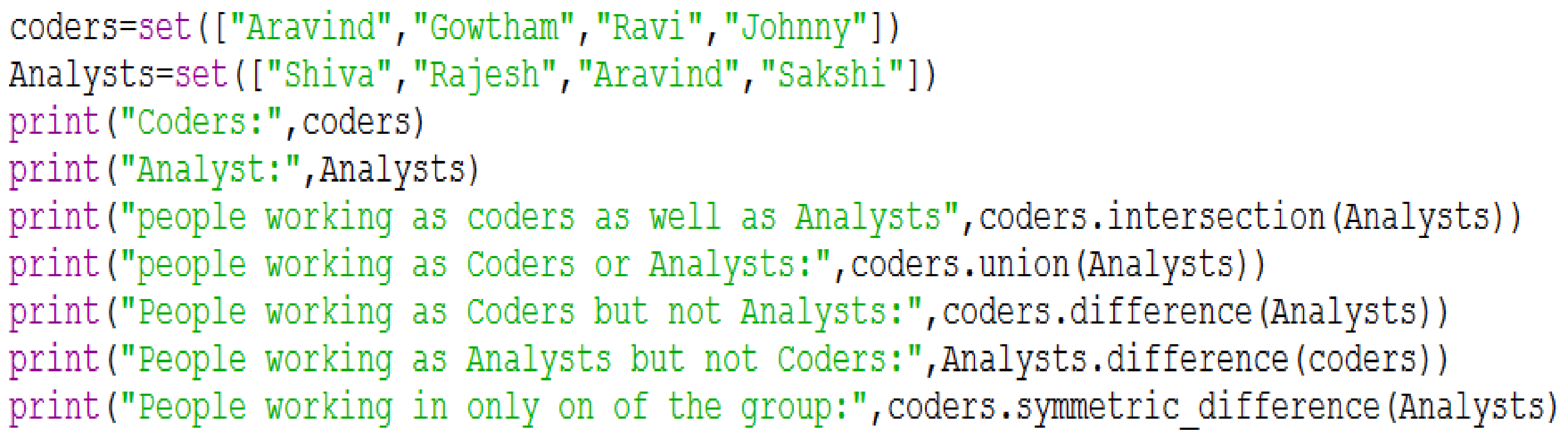
```
s1={1,2,2,3,3,3,3,3,4}
print(s1)
```

# Program to create a set



```python
list1=[1,2,3,4,5,6,5,4,3,2,1]
print("set:",set(list1))# List is converted into set type
tup1=('a','b','c','d','b','e','a')
print(set(tup1))# tuple is converted into set
str="PythonProgrammin"
print(set(str))# string is converted to set type
print((set("Hello Welcome you all for the Session ".split())))
```

#Program to find intersection,union and symmetric difference between two sets



```
coders=set(["Aravind","Gowtham","Ravi","Johnny"])
Analysts=set(["Shiva","Rajesh","Aravind","Sakshi"])
print("Coders:",coders)
print("Analyst:",Analysts)
print("people working as coders as well as Analysts",coders.intersection(Analysts))
print("people working as Coders or Analysts:",coders.union(Analysts))
print("People working as Coders but not Analysts:",coders.difference(Analysts))
print("People working as Analysts but not Coders:",Analysts.difference(coders))
print("People working in only on of the group:",coders.symmetric_difference(Analysts)
```

#Note : Two sets are equal if and only if every element in each set is contained

A set is less than another set if and only if the first set is a subset of the second set

A set is greater than another set if and only if the first set is superset of the second set

Operations on Sets

Operation	Description	Code	Output

s.update(t)	Adds element of set in the set s provided that all duplicates are avoided	s=set([1,2,3,4,5])
		t=set([6,7,8])
		s.update(t)
		print(s)
			(1,2,3,4,5,6,7,8)

s.add(x)	Adds element x to the set s provided that all duplicates are avoided	s=set([1,2,3,4,5])
		s.add(6)
		print(s)
			(1,2,3,4,5,6)

s.remove( )	Removes element x from set s. Returns KeyError if x is not present	s=set([1,2,3,4,5])
		s.remove(3)
		print(s)
			(1,2,4,5)

s.discard( )	Same as remove( ) but does not give an error if x is not present in the set	s=set([1,2,3,4,5])
		s.discard(3)
		print(s)
			(1,2,4,5)

Operation	Description	Code	Output

s.pop( )	Removes and returns any arbitrary element from s. KeyError is raised if s is empty	s=set([1,2,3,4,5])
		s.pop( )
		print(s)
			(2,3,4,5)

s.clear( )	Removes all elements from the set	s=set([1,2,3,4,5])
		s.clear ( )
		print(s)
			set{}

len(s)	Returns the length of set	s=set([1,2,3,4,5])
		print(len(s))
			5

x in s	Returns True if x is present in set s and False otherwise	s=set([1,2,3,4,5])
		print(3 in s)
			True

x not in s	Returns True if x is not present in set s and False otherwise	s=set([1,2,3,4,5])
		print(6 not in s)
			True

Operation	Description	Code	Output

s.issubset(t) or s<=t	Returns True if every element in t is present in set t and False otherwise	s=set([1,2,3,4,5])
		t=set([1,2,3,4,5,6,7,8,9])
		print(s<=t)
			True

s.issuperset(t) or s>=t	Returns True if every element in t is present in set s and False otherwise	s=set([1,2,3,4,5])
		t=set([1,2,3,4,5,6,7,8,9])
		print(s.issuperset(t))
			False

s.union(t) or s|t	Returns a set s that has elements from both set s and t	s=set([1,2,3,4,5])
		t=set([1,2,3,4,5,6,7,8,9])
		print(s|t)
			(1,2,3,4,,5,6.7.8.9)

s.intersection(t) or s&t	Returns a new set that has elements which are common to both the sets s and t	s=set([1,2,3,4,5])
		t=set([1,2,3,4,5,6,7,8,9])
		print(s&t)
			(1,2,3,4,5)

s.intersection_update(t)	Returns a set that has elements which are common to both sets s and t	s=set([1,2,3,4,5])
		t=set([1,2,3,4,5,6,7,8,9])
		s.Intersetion_update(t)
		print(s)
			(1,2,3,4,5)

Operation	Description	Code	Output

s.difference(t) or s-t	Returns a new set that has elements in set s but not in t	s=set([1,2,10)
		t=set([1,2,3,4,5,6,7,8,9])
		z=s-t
		print(z)
			10

s.difference_update(t)	Removes all elements of another set from this set	s=set([1,2,10)
		t=set([1,2,3,4,5,6,7,8,9])
		s.difference_update(t)
		print(s)
			10

s.symmetric_difference(t) or s^t	Returns a new set with elements either in s or in t but not both	s=set([1,2,10,12)
		t=set([1,2,3,4,5,6,7,8,9,10])
		z=s^t
		print(z)
			(3,4,5,6,7,8,9,12)

s.copy( )	Returns a copy of set s	s=set([1,2,10,12])
		t=set([1,2,3,4,5,6,7,8,9,10])
		print(s.copy( ))
			(1,2,12,10)

s.isdisjoint(t)	Returns True if two sets have a null intersetion	s=set([1,2,3])
		t=set([4,5,6])
		Print(s.isdisjoint(t))
			True

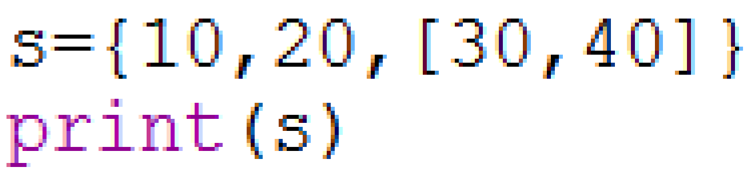
Operation	Description	Code	Output
all(s)	Returns True if all elements in the set are True and False othewise	s=set([0,1,,2,3,4]) print(all(s))	False
any(s)	Returns True if any of the elements in the set is True, Returns False if the set is empty	s=set([0,1,,2,3,4]) print(any(s))	True
enumerate(s)	Returns an enumerate object which contains index as well as value of all the items of set as a pair	s=set([‘a’,’b’,’c’,’d’]) for i in enumerate(s): print(i,end=“”)	(0,’a’)(1,’c’)((2,’b’)(‘3,’d’)
max(s)	Returns the maximum value in a set	s=set([0,1,2,3,4,5]) print(max(s))	5
min(s)	Returns the minimum value in set	s=set([0,1,2,3,4,5]) print(min(s))	0
sum(s)	Returns the sum of elements in the set	S=set([0,1,2,3,4,5]) print(sum(s))	15

Operation	Description	Code	Output

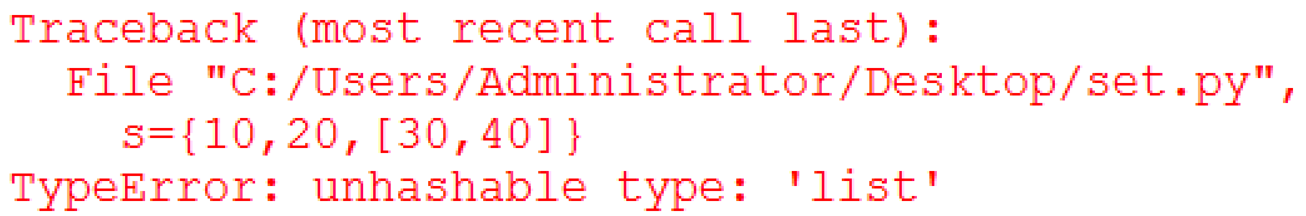
sorted(s)	Returns a new sorted list from elements in the set.	s=set([5,4,3,2,1,0])
		print(sorted(s))
			[0,1,2,3,4,5]

s==t and s!=t	s= = t returns True if the two set are equivalent and False otherwise.
	S!=t returns True if both sets are not equivalent and False otherwise
		s=set([‘a’,’b’,’c’])
		t=set(“abc”)
		z=set(tuple(‘abc’))
		print(s = = t)
		print(s!=z)
			True
			False

- A set cannot contain other mutable objects(like lists)

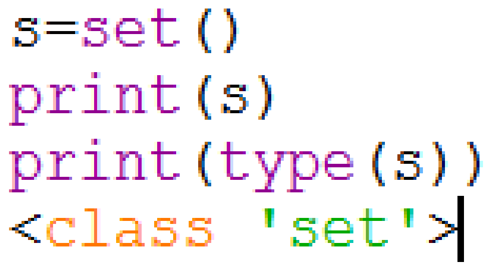


```
s={10,20,[30,40]}
print(s)
```

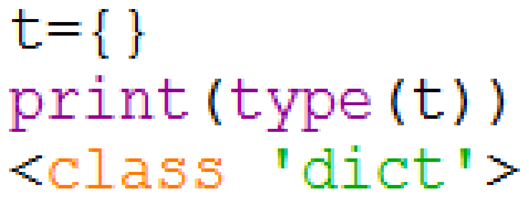


```
Traceback (most recent call last):
  File "C:/Users/Administrator/Desktop/set.py",
    s={10,20,[30,40]}
TypeError: unhashable type: 'list'
```

- To make an empty set make use of set( )

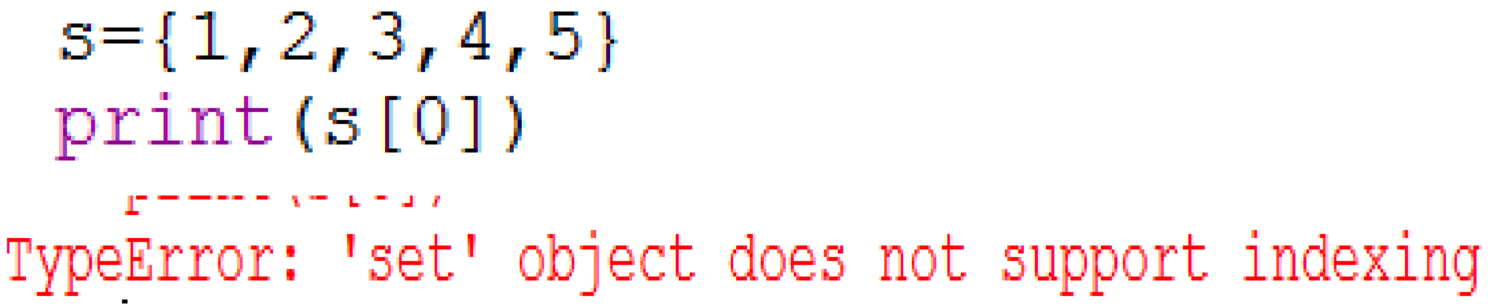


```
s=set()
print(s)
print(type(s))
<class 'set'>
```



```
t={}
print(type(t))
<class 'dict'>
```

- Since sets are unordered, indexing has no meaning over it
- Set operations do not allow users to access or change an element using indexing or slicing



```
s={1,2,3,4,5}
print(s[0])
```

```
TypeError: 'set' object does not support indexing
```

Note: A set can be created from a list but a set cannot contain a list

The copy( ) method makes a shallow copy of the set. This means that all the objects in the new set are references to the same objects as the original set

To add a single element in the set use the add( )method and to ass multiple elements in the set, use the update( ) method

Dictionaries

- Python’s dictionaries stores data as a pair key and value
- The key value pairs are enclosed with curly braces { }
- The key must be unique[Immutable], each key value pair separated from the other using a colon (**:**), and consecutive items are seperated by commas
- To access any value in the dictionary, you just need to specify its key in square braces ([ ])
- Python dictionary is an unordered collection of items
- Dictionaries are mutable i.e., it is possible to add, modify and delete key-value pairs
- Keys are used instead of indexes
- Keys are used to access elements in dictionary and keys can be of type- strings, number, list or tuple etc

- While other compound datatypes have only value as an element, a dictionary has a set of key & value pair known as item.
- Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
- We can also create a dictionary using the built-in function dict( )
- Dictionary is known as Associative Array
- Dictionary keys are case-sensitive
- **Syntax for defining a Dictionary**
  - Dictionary_name = { key_1: value_1, key_2: value_2, key_3: value_3}

```
                        or
Dictionary_name = { key_1: value_1,
                        key_2: value_2,
                        key_3: value_3}
```

Examples:

- # creation of empty dictionary my_dict = {}
- # dictionary with integer keys my_dict = {1: 'apple', 2: 'ball'}
- # dictionary with mixed keys my_dict = {'name': 'John', 1: [2, 4, 3]}
- # using dict( ) my_dict = dict({1:'apple', 2:'ball'})
-
- # from sequence having each item as a pair
  my_dict = dict([(1,'apple'), (2,'ball')])

phonebook = { } # creation of empty Dictionary
phonebook = { " Ravi": 9247448766} # Dict. with K-V pair
phonebook={"Ravi":9247448766,"Rahul":9985933931}
# Dict. With 2 K-V Pairs

- **Ex:**

```
plant={ }# we can have our own index
plant [1]="Rose"
plant[2]="Lotus"
plant["name"]="Jasmin"
plant["color"]="Green"
print(plant)
```

```
Dict={"item":"chocolate","price":100}
print(Dict["item"])
print(Dict["price"])
```

IIIT BASARA

Infosys | Campus Connect

For Details Contact Me @ :
9247448766
ravikanth27787@gmail.com

python
Programming Language