



SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai



DEPARTMENT OF CSE



19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

❖ A readable, dynamic, pleasant, flexible, fast and powerful
language



UNIT II DATA TYPES, EXPRESSIONS, STATEMENTS


- Python interpreter and interactive mode, debugging; values and types: int, float, boolean, string , and list; variables, expressions, statements, tuple assignment, precedence of operators, comments; Illustrative programs: exchange the values of two variables, circulate the values of n variables, distance between two points.



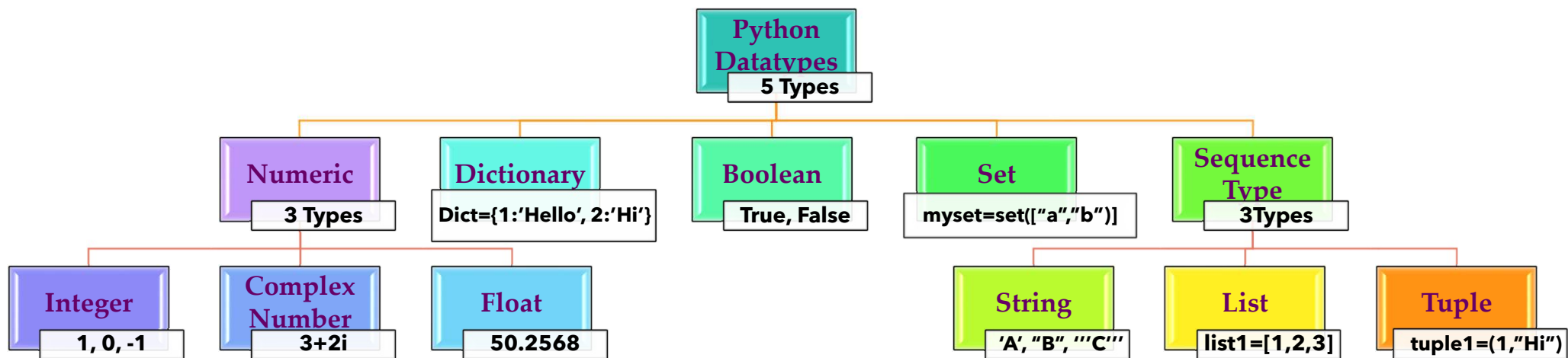


Recap

- **Values and Types**

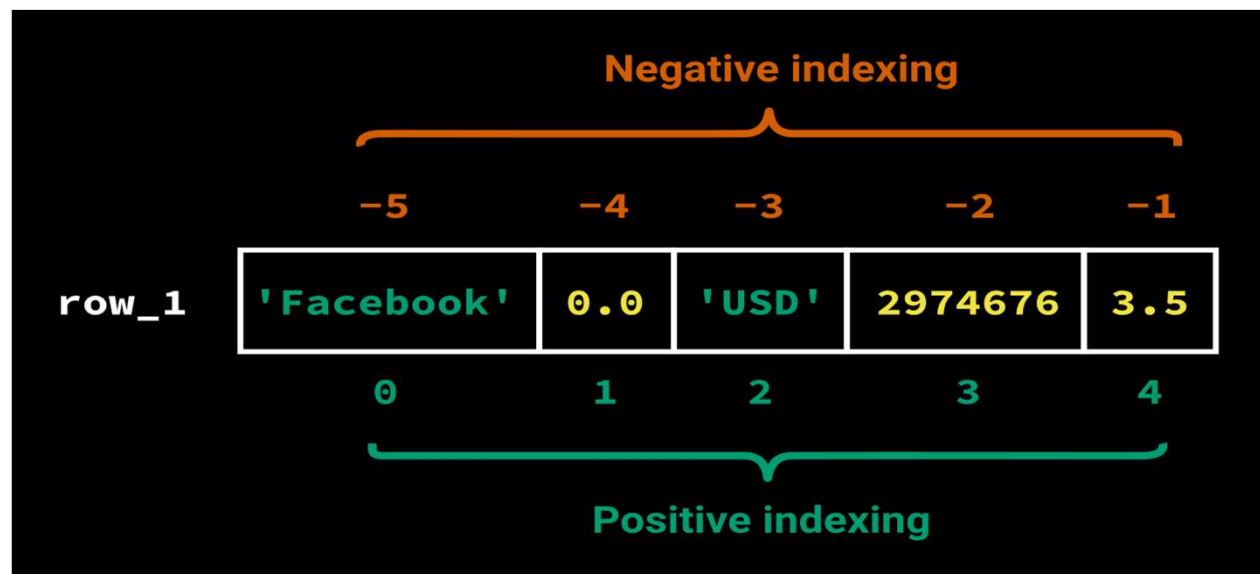
- Numeric
 - Integer
 - Float
 - Complex
 - Boolean
 - Sequence
 - String
- 

Python Datatypes



Sequence Datatype - List

- List in Python are ordered and have a definite count.
- The elements in a list are indexed according to a definite sequence and the indexing of a list is done with 0 being the first index.



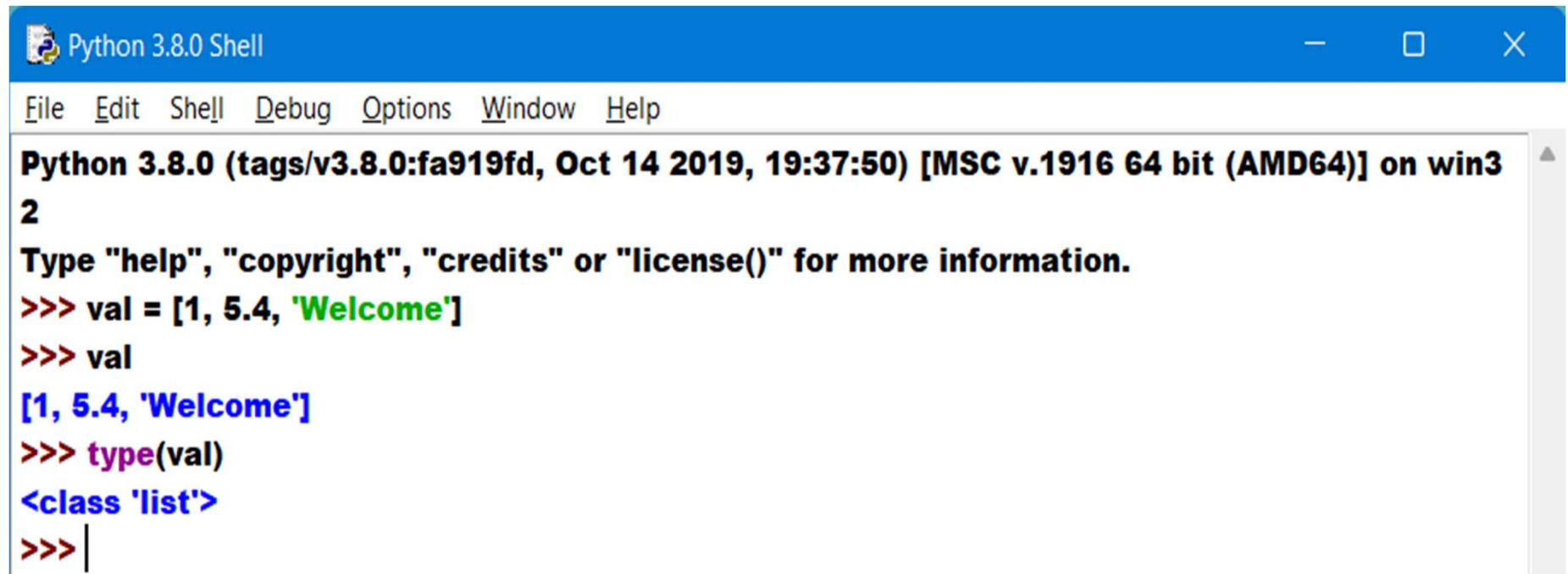


Sequence Datatype - List

- The List is a **mutable sequence of objects**, and the list class represents the List type in Python.
- Lists are **array-based sequences**, and the index starts from **zero to n-1**.
- Need to use **square brackets []** to represent a list and **comma ‘,’** to separate the elements of the list.



Sequence Datatype – List Creation

A screenshot of a Python 3.8.0 Shell window. The window has a blue title bar with the text "Python 3.8.0 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main text area shows the following content:

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> val = [1, 5.4, 'Welcome']
>>> val
[1, 5.4, 'Welcome']
>>> type(val)
<class 'list'>
>>> |
```

The code demonstrates the creation of a list named 'val' containing the elements 1, 5.4, and 'Welcome'. It then prints the list and checks its type, which is confirmed to be a list.



Sequence Datatype - List

- The list can store different types of elements.
- List can grow dynamically.

```
>>> colorSet={'Red','Blue','Green','Yellow'}
>>> fruitsTuple=('Apple','Orange','Banana')
>>> colorsList=list(colorSet)
>>> colorsList
['Blue', 'Green', 'Yellow', 'Red']
>>> fruitsList=list(fruitsTuple)
>>> fruitsList
['Apple', 'Orange', 'Banana']
```



Sequence Datatype - List

```
>>> fruitsList
['Apple', 'Orange', 'Banana']
>>> newFruitsList = ['Kiwi', 'Grapes', fruitsList]
>>> newFruitsList[0]
'Kiwi'
>>> newFruitsList[1]
'Grapes'
>>> newFruitsList[2][1]
'Orange'
>>> newFruitsList[2][2]
'Banana'
>>> |
```



Sequence Datatype - Tuple

- The Tuple is an **Immutable sequence of objects**, and the **tuple class** represents a Tuple in Python.
- A tuple is almost similar to a list except for the fact the **list is mutable**, and **tuple is immutable**.
- For Tuple also the **index starts from zero to n-1**, and we need to use **brackets ()** to represent a tuple and **comma ‘,’** to separate the elements of the tuple.



Sequence Datatype - Tuple

```
>>> numberTuple=(1,2,3,4,5)
>>> fruitsTuple=('apple','orange','banana')
>>> mixedTuple = (1, 10.5, -13.2, 'Welcome')
>>> numberTuple[1]
2
>>> fruitsTuple[2]
'banana'
>>> mixedTuple[2]
-13.2
>>> |
>>> numberTuple[5] = 6
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    numberTuple[5] = 6
TypeError: 'tuple' object does not support item assignment
>>> numberTuple[1] = 11
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    numberTuple[1] = 11
TypeError: 'tuple' object does not support item assignment
>>>
```

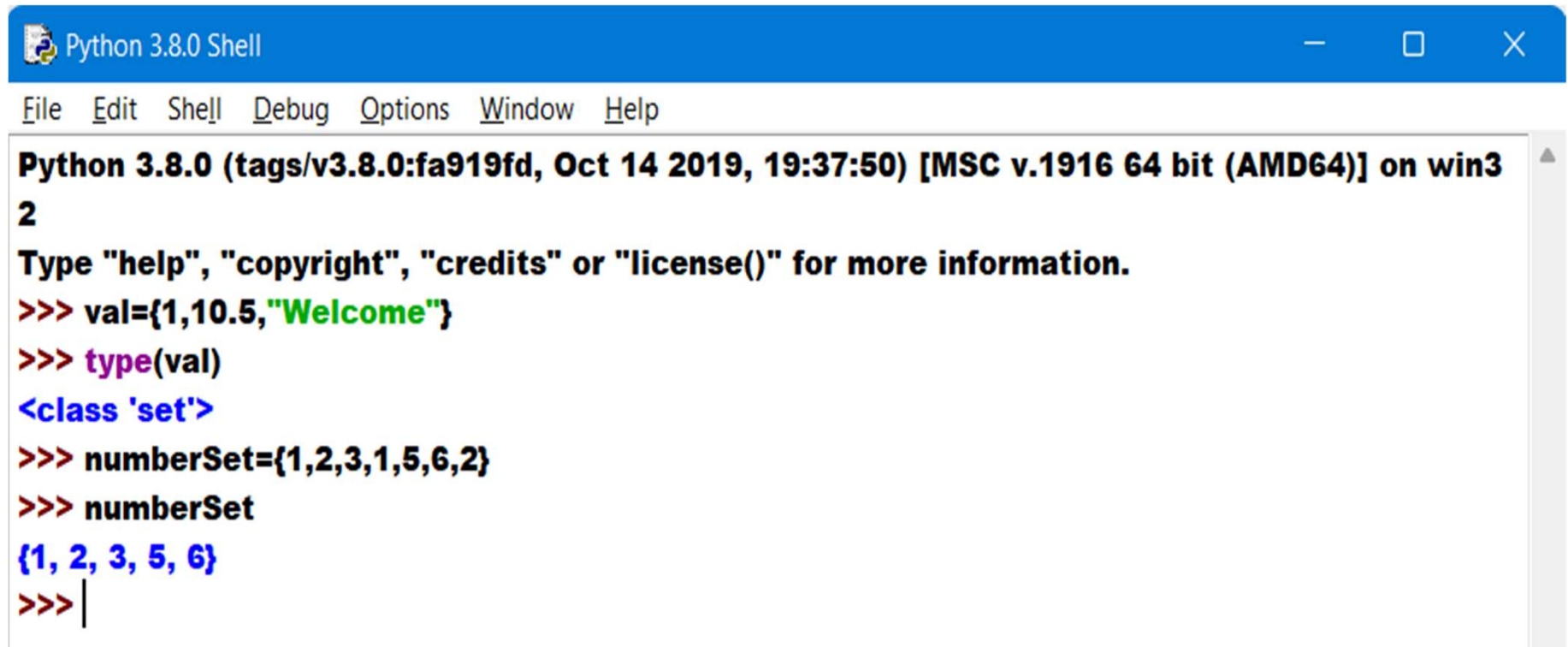


Set

- A **Set** is an **unordered mutable sequence** of **unique objects**. The **set class** represents Sets in Python.
- **Sets** are mutable, and we can **add or remove** items in a **Set**.
- Since they are **unordered**, they will **not have index positions**, and we **cannot** perform **indexing or slicing operations**.
- set are **unordered** and contain **unique elements**.



Set



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> val={1,10.5,"Welcome"}
>>> type(val)
<class 'set'>
>>> numberSet={1,2,3,1,5,6,2}
>>> numberSet
{1, 2, 3, 5, 6}
>>> |
```



Set

- Since sets are **mutable**, we are allowed to **add elements** to a set after creation.
- We can use **add()** function to add an element to a set.

```
>>> numberSet={1,2,3,1,5,6,2}
```

```
>>> numberSet
```


```
{1, 2, 3, 5, 6}
```

```
>>> numberSet.add(7)
```

```
>>> numberSet
```

```
{1, 2, 3, 5, 6, 7}
```

```
>>> |
```





FrozenSet


- A **frozenset** is an **immutable version** of a **Set**.
- They have all the properties similar to a **Set**, but you **cannot** perform any **modification operation** on a **FrozenSet**.
- **Frozenset** can be created by calling the **frozenset() function**, which takes an iterable as a parameter.
- The **frozenset** also **doesn't allow duplicate** values like a **Set**





Frozenset

```
>>> numberFrozenSet = frozenset([1, 2, 3])
>>> numberFrozenSet
frozenset({1, 2, 3})
>>> numberFrozenSet = frozenset([1, 2, 3, 1, 4, 2])
>>> numberFrozenSet
frozenset({1, 2, 3, 4})
>>> numberFrozenSet.add(11)
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    numberFrozenSet.add(11)
AttributeError: 'frozenset' object has no attribute 'add'
>>>
```





Dictionary

- The **Dictionary** or **Mapping** is a **mutable unordered set** of **key-value pairs**.
- Whenever we are adding a **key** to the dictionary, we must also add a **value** to it (which is changeable at later point).
- The **dict class** represents the dictionary in Python.
- need to use **curly braces {}** to represent a **dictionary**.
- The **key** and its **value** should be separated by a **colon ':'** and each **key-value** pair should be separated by a **comma ','**.



Dictionary

```
>>> numbers = { 1 : 'One', 2 : 'Two', 3 : 'Three'}
>>> type(numbers)
<class 'dict'>
>>> numbers[2]
'Two'
>>> numbers[1]
'One'
>>> numbers[2] = 'TTTWWWOOO'
>>> numbers[2]
'TTTWWWOOO'
>>> |
```



Dictionary

- Keys can have only **immutable hashable objects** such as **int, float, str, bool, frozenset, tuple**, but **mutable types** such as **list, set, dict** is **not permitted**.
- On the other hand, the **values** can be of **any datatype**.



Dictionary

```
>>> l=[1,2,3]
```

```
>>> s={1,2,3}
```

```
>>> type(l)
```

```
<class 'list'>
```

```
>>> type(s)
```

```
<class 'set'>
```

```
>>> test = {l : 'List'}
```

```
Traceback (most recent call last):
```

```
File "<pyshell#21>", line 1, in <module>
```

```
test = {l : 'List'}
```

```
TypeError: unhashable type: 'list'
```

```
>>> test = {s : 'Set'}
```

```
Traceback (most recent call last):
```

```
File "<pyshell#22>", line 1, in <module>
```

```
test = {s : 'Set'}
```

```
TypeError: unhashable type: 'set'
```

```
>>>
```