Chapter -3: Introduction to Python Language

Python Strings

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

```
str = 'Hello NIELIT!'

print str                 # Prints complete string
print str[0]              # Prints first character of the string
print str[2:5]            # Prints characters starting from 3rd to 5th
print str[2:]             # Prints string starting from 3rd character
print str * 2             # Prints string two times
print str + "GKP"         # Prints concatenated string
```

This will produce the following result −

```
Hello NIELIT!
H
llo
llo NIELIT!
Hello NIELIT!Hello NIELIT!
Hello NIELIT!GKP
```

Python Lists

- A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

```
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
```

This produce the following result −

['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']

Python Tuples

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are:
  - Lists are enclosed in brackets ([ ]) and their elements and size can be changed,
  - while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as **read-only** lists.

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple               # Prints complete list
print tuple[0]            # Prints first element of the list
print tuple[1:3]          # Prints elements starting from 2nd till 3rd
print tuple[2:]           # Prints elements starting from 3rd element
print tinytuple * 2       # Prints list two times
print tuple + tinytuple   # Prints concatenated lists
```

This produce the following result −

```
('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')
```

Python Dictionary

- Python's dictionaries are kind of hash table type.
- They work like associative arrays or hashes found in Perl and consist of key-value pairs.
- A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}


print dict['one']       # Prints value for 'one' key
print dict[2]           # Prints value for 2 key
print tinydict          # Prints complete dictionary
print tinydict.keys()   # Prints all the keys
print tinydict.values() # Prints all the values
```

This produce the following result −

```
This is one
This is two
{'dept': 'sales', 'code': 6734, 'name': 'john'}
['dept', 'code', 'name']
['sales', 6734, 'john']
```

Data Type Conversion

To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value.

SNo.	Function	Description
1	**int(x [,base])**	Converts x to an integer. base specifies the base if x is a string.
2	**long(x [,base])**	Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)**	Converts x to a floating-point number.
4	**complex(real [,imag])**	Creates a complex number.
5	**str(x)**	Converts object x to a string representation.
6	**repr(x)**	Converts object x to an expression string.
7	**eval(str)**	Evaluates a string and returns an object.
8	**tuple(s)**.	Converts s to a tuple
9	**list(s)**	Converts s to a list.
10	**set(s)**	Converts s to a set.
11	**dict(d)**	Creates a dictionary. d must be a sequence of (key,value) tuples.
12	**frozenset(s)**	Converts s to a frozen set.
13	**chr(x)**	Converts an integer to a character.
14	**unichr(x)**.	Converts an integer to a Unicode character
15	**ord(x)**.	Converts a single character to its integer value
16	**hex(x)**	Converts an integer to a hexadecimal string.
17	**oct(x)**	Converts an integer to an octal string.