

#3

VARIABLE TYPES



ZOOMING



Assigning Values to Variables

- It doesn't need explicit declaration to reserve memory space.
- Declaration happens automatically when we assign values to variable.

```
counter = 100 # An integer assignment
miles = 1000.0 # A floating point
name = "John" # A string
print counter
print miles
print name
```



Multiple Assignment

- Single value for multiple variables
- Multiple values for multiple variables in single assignment line.

```
a = b = c = 100
```

```
a = b = c = 1, "john2", "John
```



Standard Data Types

- Five standard datatypes
 1. Numbers
 2. String
 3. List
 4. Tuple
 5. Dictionary



Number

- It store numeric values.
- Number object is created when we assign a value to them.

```
var1 = 1  
var2 = 10
```

- To delete variable, use keyword **del**

```
del var1
```



Number

- Python support four different numeric types.
 1. **int** – signed integer.
 2. **long** – long integer, also represent octal and hexadecimal.
 3. **float** – floating point real values.
 4. **complex** – complex values.



String

- It represented inside of quotation marks.
- Quotes: single or double.
- Subset of string can be taken by using slice operator [] and [:] with indexes 0 : -1
- Concatenation operator plus (+) sign.
- Replication operator star (*) sign.



String

```
str = 'Hello World!'

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 + "TEST" # Prints concatenated string
```



#4

LIST



ZOOMING



List - Create list

```
# empty list
my_list = []
# list of integers
my_list = [1, 2, 3]
# list with mixed datatypes
my_list = [1, "Hello", 3.4]
# nested list
my_list = ["mouse", [8, 4, 6], ['a']]
```



List – Access list element

```
my_list = ['p', 'r', 'o', 'b', 'e']  
print(my_list[0]) # Output: p  
print(my_list[2]) # Output: o  
print(my_list[4]) # Output: e  
print(my_list[-1]) # Output: e  
print(my_list[-5]) # Output: p
```

```
n_list = ["Happy", [2, 0, 1, 5]]  
print(n_list[0][1]) # Output: a  
print(n_list[1][3]) # Output: 5
```



List – Access list element

P	R	O	G	R	A	M	I	Z	
0	1	2	3	4	5	6	7	8	9
-9	-8	-7	-6	-5	-4	-3	-2	-1	

```
my_list = ['p','r','o','g','r','a','m','i','z']
```

```
# elements 3rd to 5th
```

```
print(my_list[2:5])
```

```
# elements beginning to 4th
```

```
print(my_list[:-5])
```

```
# elements 6th to end
```

```
print(my_list[5:])
```

```
# elements beginning to end
```

```
print(my_list[:])
```



List – Change or Add elements

```
# mistake values
```

```
odd = [2, 4, 6, 8]
```

```
odd[0] = 1 # change the 1st item
```

```
print(odd) # Output: [1, 4, 6, 8]
```

```
odd[1:4] = [3, 5, 7]
```

```
print(odd) # Output: [1, 3, 5, 7]
```

```
odd = [1, 3, 5]
```

```
odd.append(7)
```

```
print(odd)
```

```
# Output: [1, 3, 5, 7]
```

```
odd.extend([9, 11, 13])
```

```
print(odd)
```

```
# Output: [1, 3, 5, 7, 9, 11, 13]
```



List – Delete elements

```
my_list = ['p','r','o','b','l','e','m']
```

```
del my_list[2] # delete one item
```

```
print(my_list) # Output: ['p', 'r', 'b', 'l', 'e', 'm']
```

```
del my_list[1:5] # delete multiple items
```

```
print(my_list) # Output: ['p', 'm']
```

```
del my_list # delete entire list
```

```
print(my_list) # Error: List not defined
```



List Method – **append()**

- It is used to add elements to the last position of List.

```
List = ['Mathematics', 'chemistry', 1997, 2000]  
List.append(20544)  
print(List)
```

```
['Mathematics', 'chemistry', 1997, 2000, 20544]
```



List Method – insert()

- It is used to insert element at specific position.

```
List = ['Mathematics', 'chemistry', 1997, 2000]  
List.insert(2, 10087)  
print(List)
```

```
['Mathematics', 'chemistry', 10087, 1997, 2000]
```



List Method – extend()

- Add multiple values or another list into the end of the current list

```
List1 = [1, 2, 3]  
List2 = [2, 3, 4, 5]
```

```
# Add List2 to List1  
List1.extend(List2)  
print(List1)
```

```
#Add List1 to List2 now  
List2.extend(List1)  
print(List2)
```

```
[1, 2, 3, 2, 3, 4, 5]  
[2, 3, 4, 5, 1, 2, 3, 2, 3, 4, 5]
```



List Method – **sum(), count(), len(), min(), max()**

```
List = [1, 2, 3, 4, 5 , 1]
print(sum(List)) //16
print(List.count(1)) //2
print(len(List)) //6
print(min(List)) //1
print(max(List)) //5
```



List Method – **sort(), reverse()**

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
```

```
#Reverse flag is set True
```

```
List.sort(reverse=True)
```

```
print(List)
```

```
[5.33, 4.445, 3, 2.5, 2.3, 1.054]
```

```
sorted(List)
```

```
print(List)
```

```
[1.054, 2.3, 2.5, 3, 4.445, 5.33]
```



List Method – **pop()**, **del()**, **remove()**

pop(): Index is not a necessary parameter, if not mentioned takes the last index.

del() : Element to be deleted is mentioned using list name and index.

remove(): Element to be deleted is mentioned using list name and element.

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
```

```
print(List.pop())
```

```
2.5
```

```
print(List.pop(0))
```

```
2.3
```

```
del List[0]
```

```
[4.445, 3, 5.33, 1.054, 2.5]
```

```
print(List)
```

```
print(List.remove(3))
```

```
[4.445, 5.33, 1.054, 2.5]
```



#5

TUPLES



ZOOMING



Tuples

- It is a collection of object much like a list.
- The main different between tuple and list is that tuples are **immutable**.
- It represent as ().
- Values of a tuple are syntactically separated by commas.
- Tuple elements cannot be changes.



Tuples - Create

```
Tuple1 = () //empty tuple
```

```
Tuple2 = ('zooming', 'For') //tuple with strings
```

```
list1 = [1, 2, 4, 5, 6]
```

```
Tuple3 = tuple(list1) //tuple with the use of list
```

```
Tuple4 = ('zooming',) * 3 //tuple with repetition
```

```
Tuple5 = (5, 'Welcome', 7, 'zooming') //tuple with mixed datatypes
```



Tuples - Concatenation

- Concatenation of tuple is the process of joining of two or more Tuples.
- Concatenation is done by the use of '+' operator.
- Concatenation of tuples is done always from the end of the original tuple.
- Other arithmetic operations do not apply on Tuples.



```
# Concatenation of tuples
Tuple1 = (0, 1, 2, 3)
Tuple2 = ('Zooming', 'For', 'stud')

Tuple3 = Tuple1 + Tuple2

print("\nTuples after Concatenation: ")
print(Tuple3)
```

```
Tuples after Concatenation:
(0, 1, 2, 3, 'Zooming', 'For', 'stud')
```


Tuples - Slicing

```
# with Numbers
Tuple1 = tuple('ZOOMING')

# From First element
print(Tuple1[1:])

# Reversing the Tuple
print(Tuple1[::-1])

# Printing elements of a Range
print(Tuple1[2:5])
```

```
('O','O','M','I','N','G')
('G','N','I','M','O','O','Z')
('O','M','I')
```



Tuples - Deleting

- Tuples are immutable and hence they do not allow deletion of a part of it.
- Entire tuple gets deleted by the use of `del()` method.
- **Note-** Printing of Tuple after deletion results to an Error.

```
# Deleting a Tuple
```

```
Tuple1 = (0, 1, 2, 3, 4)
```

```
del Tuple1
```

```
print(Tuple1)
```

NameError: name 'Tuple1' is not defined



Tuple – Built-in-Methods

BUILT-IN FUNCTION	DESCRIPTION
all()	Returns true if all element are true or if tuple is empty
any()	return true if any element of the tuple is true. if tuple is empty, return false
len()	Returns length of the tuple or size of the tuple
enumerate()	Returns enumerate object of tuple
max()	return maximum element of given tuple
min()	return minimum element of given tuple
<u>sum()</u>	Sums up the numbers in the tuple
<u>sorted()</u>	input elements in the tuple and return a new sorted list
<u>tuple()</u>	Convert an iterable to a tuple.



#6

DICTIONARY



ZOOMING



Dictionary

- It is an unordered collection of data values, used to store data values like a map.
- Dictionary holds **key:value** pair.
- Key value is provided in the dictionary to make it more optimized.
- Each key-value pair in a Dictionary is separated by a colon :,
 - whereas each key is separated by a 'comma'.
- Key – must be unique and immutable datatype.
- Value – can be repeated with any datatype..



Dictionary - Create

Creating an empty Dictionary

```
Dict = {}
```

Creating a Dictionary with Integer Keys

```
Dict = {1: 'Zooming', 2: 'For', 3: 'Zooming'}
```

Creating a Dictionary with Mixed keys

```
Dict = {'Name': 'Zooming', 1: [1, 2, 3, 4]}
```

Creating a Dictionary with dict() method

```
Dict = dict({1: 'Zooming', 2: 'For', 3: 'Zooming'})
```

Creating a Dictionary with each item as a Pair

```
Dict = dict([(1, 'Zooming'), (2, 'For')])
```



Dictionary – Adding element

```
# Creating an empty Dictionary  
Dict = {}
```

```
# Adding elements one at a time  
Dict[0] = 'Zooming'  
Dict[2] = 'For'  
Dict[3] = 1
```

```
# Adding set of values  
# to a single Key  
Dict['Value_set'] = 2, 3, 4
```

```
# Updating existing Key's Value  
Dict[2] = 'Welcome'
```

```
# Adding Nested Key value to Dictionary  
Dict[5] = {'Nested' : {'1' : 'Life', '2' : 'Zooming'}}
```



Dictionary – Accessing element

```
# Creating a Dictionary
Dict = {1: 'Zooming', 'name': 'For', 3: 'Zooming'}

# accessing a element using key
print("Accessing a element using key:")
print(Dict['name'])

# accessing a element using key
print("Accessing a element using key:")
print(Dict[1])

# accessing a element using get()
# method
print("Accessing a element using get:")
print(Dict.get(3))
```



Dictionary – Removing element

```
Dict = { 5 : 'Welcome', 6 : 'To', 7 : 'Geeks',  
        'A' : {1 : 'Geeks', 2 : 'For', 3 : 'Geeks'},  
        'B' : {1 : 'Geeks', 2 : 'Life'}}  
print("Initial Dictionary: ")  
print(Dict)
```

```
# Deleting a Key value  
del Dict[6]
```

```
# Deleting a Key from Nested Dictionary  
del Dict['A'][2]
```

```
# Deleting a Key using pop()  
Dict.pop(5)
```

```
# Deleting entire Dictionary  
Dict.clear()
```



METHODS	DESCRIPTION
<u>copy()</u>	They copy() method returns a shallow copy of the dictionary.
<u>clear()</u>	The clear() method removes all items from the dictionary.
<u>pop()</u>	Removes and returns an element from a dictionary having the given key.
<u>popitem()</u>	Removes the arbitrary key-value pair from the dictionary and returns it as tuple.
<u>get()</u>	It is a conventional method to access a value for a key.
<u>dictionary_name.values()</u>	returns a list of all the values available in a given dictionary.
<u>str()</u>	Produces a printable string representation of a dictionary.
<u>update()</u>	Adds dictionary dict2's key-values pairs to dict
<u>setdefault()</u>	Set dict[key]=default if key is not already in dict
<u>keys()</u>	Returns list of dictionary dict's keys
<u>items()</u>	Returns a list of dict's (key, value) tuple pairs
<u>has_key()</u>	Returns true if key in dictionary dict, false otherwise
<u>fromkeys()</u>	Create a new dictionary with keys from seq and values set to value.
<u>type()</u>	Returns the type of the passed variable.
<u>cmp()</u>	Compares elements of both dict.

