Mastering Python Training

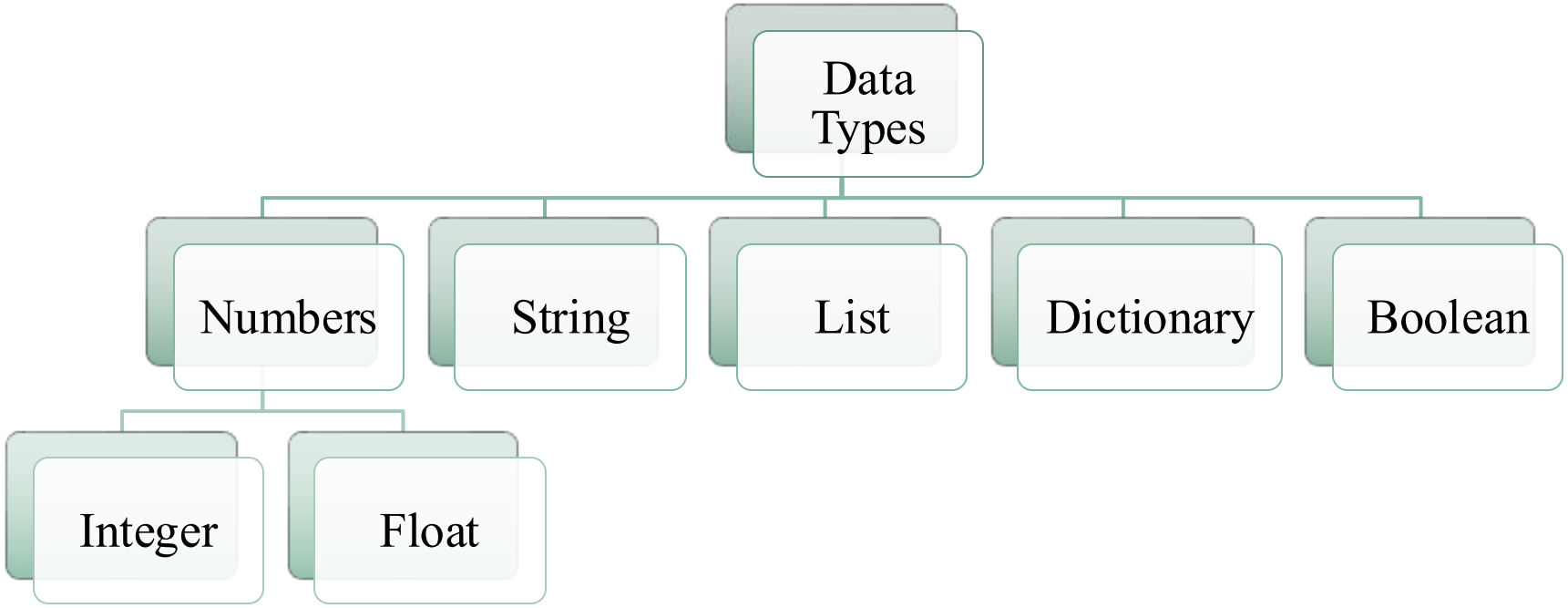
PRESENTED BY:-

ENG. MAHA MOHY EL-DIN

Agenda

- Data Types
  - Integer
  - Float
  - String
- String methods
- Comments
- String formatting
- Arithmetic operators
- Lists
- List methods
- Tuples
- Set Methods

Data Types


Data Types
Numbers
String
List
Dictionary
Boolean
Integer
Float

Basic Syntax Rules

- Variable name starts with characters or _.
- Variable name can not include special characters.
- Variable name can not start with numbers or special characters.

Data Types(Numbers)

- **Integer**

```
X=10
Print (x)
>> 10
```

- **Float**

```
X=10.6
Print (x)
>> 10.6
```

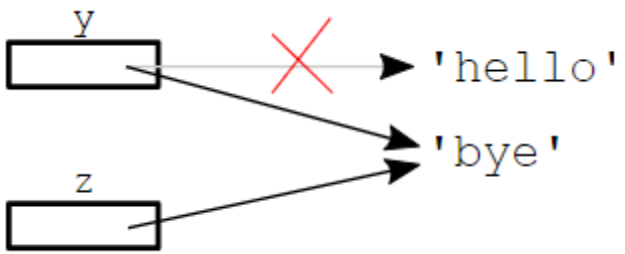
Data Types(Strings)

- Strings could be written between ‘ ’, or “ ”.
- MyString= 'I Love Python'.
- MyString= "I Love Python." .

Quick Quiz

- What will be the output of the below example?

```
y = ‘hello’
y = ‘bye’
z = y
print (y)
print (z)
```



Data Types(Strings)

- **String Slicing**
- The 1st element has an index= 0.
- The last element has an index= -1.

```
MyString= 'I Love Python'
print (MyString[0])
>> I
```

```
MyString= 'I Love Python'
print (MyString[-1])
>> n
```

Data Types(Strings)

String Slicing

- To access multiple items write [Start:End], end not included.
- [Start:End:steps].

```
MyString= 'I Love Python'
print (MyString[0:6])
>> I Love
```

```
MyString= 'I Love Python'
print (MyString[:10])
>> I Love Pyt
```

```
MyString= 'I Love Python'
print (MyString[5:])
>> e Python
```

- To print the full data write print (MyString[:])

String Methods

- Len(): Retrieves the length of the string, which means the total number of characters including spaces.

```
MyString= 'I Love Python'
print (len(MyString))
>> 13
```

```
MyString= '   I Love Python    '
print (len(MyString))
>> 19
```

String Methods

- strip(): Removes spaces to the right, and the left of the string.
- rstrip(): Removes spaces to the right of the string.
- lstrip(): Removes spaces to the left of the string.

```
x = "      I Love Python. "
print (x.strip())
>> I Love Python.
```

```
x = "      I Love Python.   "
print (x.rstrip())
>>      I Love Python.
```

```
x =  "   I Love Python.   "
print (x.lstrip())
>> I Love Python.
```

Quick Quiz

- How to delete the $ before, and after the string.

```
x =  “$$$I Love Python$$$.”
```

```
x =  '$$I Love Python$$$'
print (x.strip("$"))
>> I Love Python.
```

String Methods

- upper()

```
MyString= 'I Love Python'
print (MyString.upper())
>> I LOVE PYTHON
```

- lower()

```
MyString= 'I Love Python'
print (MyString.lower())
>> i love python
```

Email Slicing Example

- If I have an email address, and want to get the username and the domain, here are the steps:
- To get the username:

```
Email= "Maha.Mohy@gmail.com"
 print (Email[:Email.index("@")])
>>Maha.Mohy
```

- To get the domain:

```
Email= "Maha.Mohy@gmail.com"
 print (Email[Email.index("@")+1:])
>>gmail.com
```

Comments

- Useful when your code needs further explanation. Either for your future self and anybody else.
- Useful when you want to remove the code from execution but not permanently.
- Comments in Python are done with #.
- Write ctrl+/ to add multiple line comments, and to uncomment it.

String Methods

- split(): Retrieves a list of all the items.

```
a= "I Love Python and Java and Dart"
print (a.split())
>> ['I', 'Love', 'Python', 'and', 'Java', 'and', 'Dart']
```

- What if we replaced the spaces between words with -?

```
a= "I-Love-Python-and-Java-and-Dart"
print (a.split("-"))
>> ['I', 'Love', 'Python', 'and', 'Java', 'and', 'Dart']
```

String Methods

- Max Split: To split only a defined number of elements.

```
a= "I Love Python and Java and Dart"
print (a.split(' ',3))
>> ['I', 'Love', 'Python', 'and Java and Dart']
```

- rsplit(): Split the elements, starting from right.

```
a= "I Love Python and Java and Dart"
print (a.rsplit(‘ ',2))
>> ['I Love Python and Java', 'and', 'Dart']
```

String Methods

- Center(): Adds characters before and after the string.

```
a= 'Maha'
print (a.center(8)  #Spaces
>>  Maha
```

```
a= 'Maha'
print (a.center(8, '@'))  #@
>>@@Maha@@
```

String Methods

- Count(): This method counts how many times a character or a word is repeated.

```
a= 'Maha'
print (a.count("a"))
>>2   #The character a is repeated 2 times
```

```
a= ‘Ahmed Loves Programming languages like
Python,Dart because Python is easy'
print (a.count('Python'))
>>2   #Python is repeated 2 times
```

String Methods

- Count(‘char’, start,end)

```
a= 'Maha Loves Programming languages like Python,Dart because Python is easy'
print (a.count('a',0,30))
>>5   #The character a is repeated 5 times within characters 0:30
```

String Methods

- Swapcase(): Replaces upper case letters with lower case ones, and vice versa.

```
a = ' I Love Python'
print (a.swapcase())
>>I lOVE pYTHON
```

```
a = ' i lOVE pYTHON'
print (a.swapcase())
>>I Love Python
```

String Methods

- startswith(): A method that returns a Boolean value, as an answer to the question if the string starts with a defined character or not.

```
a = 'I Love Python'
print (a.startswith('I'))
>>True
```

```
a = 'I Love Python'
print (a.startswith('i'))
>>False
```

- endswith(): A method that returns a Boolean value, as an answer to the question if the string ends with a defined character or not.

```
a = 'I Love Python'
print (a.endwith('I'))
>>False
```

```
a = 'I Love Python'
print (a.endwith('e', 0,6))
>>True
```

String Methods

- index(): Returns the index of a specified character.

```
a = 'I Love Python'
print (a.index('v'))
>>4
```

```
a = 'I Love Python'
print (a.index('o', 7,12))
>>11
```

```
a = 'I Love Python'
print (a.index('P',0,5))
>>Error
```

- find(): The same as index() method, the only difference is that when selecting a character to search for and it is out of the defined range, it returns -1 instead of error.

```
a = 'I Love Python'
print (a.find('P',0,5))
>>-1
```

String Methods

- **rjust():** Right justify the text and adds spaces or special characters to the left.

```
a = 'I Love Python'
print (a.rjust(15)
>>   I Love Python
```

```
a = 'I Love Python'
print (a.rjust(15, '#')
>>##I Love Python
```

- **ljust():** Left justify the text and adds spaces or special characters to the right.

```
a = 'I Love Python'
print (a.ljust(15, '#')
>>I Love Python##
```

String Methods

- splitlines(): Returns a list of sentences written in multi line string.

```
a = “ “ “I Love Python
And Java
And Dart” ” ”
print (a.splitlines())
>>[‘I Love Python’, ‘And Java’, ‘And Dart’]
```

```
a = ‘I Love Python\nAnd Java\nAnd Dart’
print (a.splitlines())
>>[‘I Love Python’, ‘And Java’, ‘And Dart’]
```

String Methods

- replace(): Replaces the original value with the new defined value.

```
a = ‘A a B b C c D d AA'
print (a.replace('A', ‘z')
>>‘z a B b C c D d zz'
```

```
a = ‘A a B b C c D d AA'
print (a.replace('A', ‘z‘, 1)
>>‘z a B b C c D d AA'
```

String Methods

- join(): Concatenates the items in the list into a string.

```
MyList= [“I”, “Love”, “Python”]
print (“ ”.join(MyList))
>> I Love Python
```

String formatting

- format(): Returns a formatted text based on the passed parameter.

```
txt = "My name is {}, I'm {} years old".format("Maha",32)
print (txt)
>>My name is Maha, I'm 32 years old.
```

```
MyName= 'Maha'
Age=32
print (f'My name is:{MyName}, and my age is: {Age}.')
>>My name is Maha, and my age is 32.
```

For More details about formatting, please visit https://pyformat.info/

Arithmetic operations

Task	Operation
Addition	x+y
Subtraction	x-y
Multiplication	x*y
Division	x/y
Floor Division	x//y
Modulus	x%y
Power of	x**y

Lists

- List items are enclosed in square brackets.
- Lists are ordered, zero based indexing to access items.
- Lists are mutable (Add,Edit,Delete).
- List items are not unique.
- Lists can have different data types.

Lists

- MyList=[1, ‘Two’, False, 53,0.45]

```
MyList=[1, ‘Two’, False, 53, 0.45]
print (MyList)
>> [1, ‘Two’, False, 53,.45]
```

```
MyList=[1, ‘Two’, False, 53, 0.45]
print (MyList[0])
>> 1
```

```
MyList=[1, ‘Two’, False, 53, 0.45]
print (MyList[-1])
>> 0.45
```

```
MyList=[1, ‘Two’, False, 53, 0.45]
print (MyList[0:3])
>> [1, ‘Two’, False]
```

Lists (Edit items)

- We can edit the item values as following:-

```
MyList=[1, 'Two', False, 53, 0.45]
MyList [0]= -100
print (MyList)
>> [-100, 'Two', False, 53,.45]
```

- To edit multiple items:

```
MyList=[1, 'Two', False, 53, 0.45]
MyList [0:2]= [-100, 'Hello']
print (MyList)
>> [-100, 'Hello', False, 53,.45]
```

Quick Quiz

- Passing only one parameter, does not mean replacing it is just edit.

```
MyList=[1, 'Two', False, 53, 0.45]
MyList [0:2]= ['A']
print (MyList)
>> ['A', False, 53,.45]
```

Lists(Delete items)

- To delete elements use []:

```
MyList=[1, 'Two', False, 53, 0.45]
MyList [0:2]= []
print (MyList)
>> [ False, 53,.45]
```

Lists (Add items)

- append(): Adds the element at the end of the list.

```
MyList=[1, 'Two', False]
MyList.append('ABC')
print (MyList)
>> [1, 'Two', False, 'ABC']
```

- insert(): Adds the element at a specified index of the list.

```
MyList=[1, 'Two', False]
MyList.insert(1,'ABC')
print (MyList)
>> [1, 'ABC', 'Two', False]
```

List methods

- extend() : It makes a concatenation for two lists.

```
a=[1, 2,3]
b= [‘One’, ‘Two’, ‘Three’]
a.extend(b)
print (a)
>> [1, 2, 3, ‘One’, ‘Two’, ‘Three’]
```

- remove(): Removes the first occurrence of the element.

```
x=[1,2,3,1,0]
x.remove(1)
print(x)
>> [ 2, 3, 1,0]
```

List methods

- sort(): Sorts the list elements ascendingly.

```
x=[1,2,3,10,0,-50]
x.sort()
print(x)
>> [-50,0,1,2,3,10]
```

- sort(reverse=True): Sorts the list elements descending.

```
x=[1,2,3,10,0,-50]
x.sort(reverse=True)
print(x)
>> [10,3,2,1,0,-50]
```

List methods

- reverse(): This method reverses the order of the list, regardless of the alphabetical sorting.

```
x=[193,2,3,'Maha',10,0,-50]
x.reverse()
print(x)
>> [-50,0,10,'Maha',3,2,193]
```

List methods

- clear(): To clear the list and make it empty.

```
x=[1,2,3]
x.clear()
print(x)
>>[]
```

- copy(): Makes a copy of the main list.

```
x=[1,2,3]
y=x.copy()
print(x)
print (y)
>>[1,2,3]
>>[1,2,3]
```

```
x=[1,2,3]
y=x.copy()
x.append(4)
print(x)
print (y)
>>[1,2,3,4]
>>[1,2,3]
```

List methods

- count(): Counts how many times an element in the list is repeated.

```
x=[1,2,3,0,5,6,0]
print(x.count(0))
>> 2
```

- index()

```
x=['a', 'b', 'c', 'd', 'a']
print(x.index('a'))
>> 0
```

Tuple

- Tuples are the same as lists but enclosed by parentheses, the only difference is that tuples are immutable.
- The parentheses could be removed.
- Tuple items are not unique.
- Tuple could contain different data types.

```
x=('a', 'b', 'c', 'a', 1)
print(x)
>> ('a', 'b', 'c' , 'a', 1)
```

```
x= 'a', 'b', 'c'
 print(x)
>> ('a', 'b', 'c')
```

Tuple indexing

```
x=('a', 'b', 'c')
print(x[1])
>> 'b'
```

```
x=('a', 'b', 'c')
print(x[-1])
>> 'c'
```

Tuple

- Tuple is immutable, which means you can not assign a new value to the tuple or edit it.

```
x=(1,2,3,1,0)
 x.index[1]=5
 print(x)
>> TypeError: object does not
support item assignment
```

Tuple

- Tuple concatenation could be done through +.

```
x=(‘a’, ‘b’, ‘c’)
y= (1,2)
z= x+y
print(z)
>> ‘a,b,c,1,2’
```

Tuple, List, String Repeat

- To repeat tuple, List, or string use *.

```
MyTuple= (‘a’, ‘b’, ‘c’)
MyList= [1,2]
MyString= “Hello World”
```

Statement	Output
print(MyTuple * 2)	(‘a’, ‘b’, ‘c’, ‘a’, ‘b’, ‘c’)
print(MyList * 4)	[1,2,1,2,1,2,1,2,1,2]
Print (MyString *3)	Hello WorldHello WorldHello World

Tuple methods

- Count(): Counts how many times an item is repeated in the tuple.

```
MyTuple= (‘a’, ‘b’, ‘c’, ‘a’)
print(MyTuple.count(‘a’))
>> 2
```

- Index(): Retrieves the position of an element.

```
MyTuple= (‘a’, ‘b’, ‘c’)
print(MyTuple.index(‘a’))
>>0
```

Quick Quiz

- In the previous slide, how to make a concatenation to write << The position of index is: 0.

Set

- Set items are enclosed in curly braces.
- Set items are not ordered or indexed.
- Set indexing, and slicing can not be done.
- Set items are unique.

```
MySet= {‘Ali’, ‘Ahmed’, ‘Mohamed’, ‘Youssef’}
print(MySet)
```

Set methods

- Clear(): To remove all the elements, and make the set empty.

```
MySet= {‘a’, ‘b’, ‘c’, ‘a’}
MySet.clear()
print(MySet)
>> set ()
```

- Union():

```
x= {‘a’, ‘b’, ‘c’}
y= {1,2}
print(x.union(y))
>> {‘a’, ‘b’, 1,2, ‘c’}
```

```
x= {‘a’, ‘b’, ‘c’}
y= {1,2}
print(x|y)
>> {‘a’, ‘b’, 1,2, ‘c’}
```

Set methods

- Add(): This method add only one element at once.

```
x= {‘a’, ‘b’, ‘c’}
x.add(1)
print(x)
>> {‘a’, ‘b’, ‘c’, 1}
```

- Copy(): Takes a copy of the set to another set.

```
x= {‘a’, ‘b’, ‘c’}
y=x.copy()
print(x)
Print (y)
>> {‘a’, ‘b’, ‘c’}
>> {‘a’, ‘b’, ‘c’}
```

Set methods

- Remove(): Removes an element from the set.

```
x= {1,2,3}
x.remove(1)
print(x)
>> {2,3}
```

- Discard: The same as remove, but if an element is not in the set it does not return error.

```
x= {1,2,3}
x.remove(1)
x.remove(6)
print(x)
>> {2,3}
```