Mastering Python

الدرس #3

Strings, List, Tuple, Dictionary & Sets

By:

Hussam Hourani

V1.0 - NOV 2019

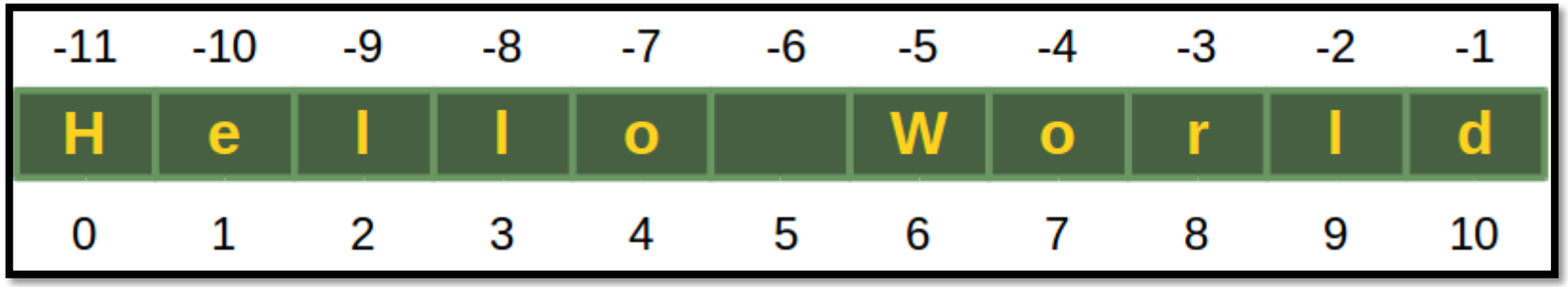
Agenda

- Strings
- List Overview
- Tuple Overview
- Sets Overview
- Dictionary Overview

- السلاسل
- القوائم
- الصفوف
- المجموعات
- القواميس

String

Strings in Python are identified as a contiguous set of characters represented in the quotation marks.



https://www.python-course.eu/python3_variables.php

```
str1 = 'Hello World!'

print (str1 )          # Prints complete string
print (str1[0])        # Prints first character of the string
print (str1[-1])
print (str1[2:5])      # Prints characters starting from 3rd to 5th
print (str1[2:])       # Prints string starting from 3rd character
print (str1 * 2)       # Prints string two times
print (str1 + "TEST") # Prints concatenated string)
```

Output

```
Hello World!
H
d
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

String

```
var1 = 'Hello World!'
var2 = "Python Programming"
print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5])
print("count(m) :", var2.count('m'))

for c in 'Hello': print(c)




print (" len(var1) :", var1, len(var1) )
str1 = "Hello" + ' ' + var2
print("str1",str1)
str2 = '%s %s %d' % ("Hello", str1, 12)
print ( "str2: ",str2)
```

Output

```
var1[0]:  H
var2[1:5]:  ytho
count(m) : 2
H
e
l
l
o
 len(var1) : Hello World! 12
str1 Hello Python Programming
str2:  Hello Hello Python Programming 12
```

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

Output

```
I want 3 pieces of item 567 for 49.95 dollars.
```

String

```
s = "hello"
print ("capitalize : ", s.capitalize()  )
print ("upper : ",  s.upper()        )
print ("center : ",  s.center(20)      )
print ("replace : ",  s.replace('l', '(ell)'))
print ("strip :",  '  world '.strip()  )
```

Output

```
capitalize :  Hello
upper :  HELLO
center :          hello
replace :  he(ell)(ell)o
strip : world
```

```
para_str = """this is a long string that is
made up of
several lines and non-printable characters such
as
TAB ( \t ) and they will show up that way when
displayed.
NEWLINEs within the string, whether explicitly
given like
this within the brackets [ \n ], or just a
NEWLINE within
the variable assignment will also show up.\a
"""
print ("paragraph string :", para_str)
```

Output

```
paragraph string : this is a long string that
is made up of
several lines and non-printable characters
such as
TAB (    ) and they will show up that way when
displayed.
NEWLINEs within the string, whether explicitly
given like
this within the brackets [
 ], or just a NEWLINE within
the variable assignment will also show up.
```

String - Practice

- Assume the following string s1=”Hello Orange Academy”
  - print (s1)
  - print (s1[0])
  - print (s1[-2])
  - print (s1[2:10])
  - print (s1[5:])
  - print (s1 * 2)
  - print (s1.capitalize())
  - print (s1.upper())
  - print (s1.center(20))
  - print (s1.replace(‘Orange', ‘Amman'))
  - print (' world '.strip())
  - s2='#'.join(['hello','Orange'])
  - print(s2)

Lists

- Lists are mutable sequences of values

```
my_list = [1,20,100,5,3,20]
print (my_list)
```

Output

```
0,1,2,3,4,5 : Index of the List
[1, 20, 100, 5, 3, 20]
-6,-5,-4,-3,-2,-1 : Index of the List
```

```
my_list1 = ["Apple",1,1.2222]
my_list2 = [10,20,["Orage",3]]
my_list3= my_list1 + my_list2
print(my_list3)
print("Apple" in my_list3 )
```

Output

```
['Apple', 1, 1.2222, 10, 20, ['Orage', 3]]
True
```

```
print(my_list3[1])
print(my_list3[5][1])
print(my_list3[2:])
print(my_list3[:-1])
```

Output

```
1
3
[1.2222, 10, 20, ['Orage', 3]]
['Apple', 1, 1.2222, 10, 20]
```

```
for item in my_list3:
        print(item)
```

Output

```
Apple
1
1.2222
10
20
['Orage', 3]
```

Lists

```
my_list = ["Apple",1] * 2
print(my_list)
```

Output

```
['Apple', 1, 'Apple', 1]
```

```
my_list = [100,2,30,60,20,15,1]
print( len(my_list) )
print( min(my_list) )
print( max(my_list) )
print( sum(my_list) )
print( sorted(my_list) )
```

Output

```
7
1
100
228
[1, 2, 15, 20, 30, 60, 100]
```

```
my_list1 = ["b","c","a"]
my_list2 = ["e","d"]
my_list2.append("f")
my_list1.extend(my_list2)
print ( my_list1 )
print ( my_list2 )
my_list1.sort()
print ( my_list1 )
```

Output

```
['b', 'c', 'a', 'e', 'd', 'f']
['e', 'd', 'f']
['a', 'b', 'c', 'd', 'e', 'f
```

```
my_list1 = ["b","c","a"]
my_list1.pop()            # or pop( Item Index)
print (my_list1)
```

Output

```
['b', 'c']
```

Lists

```
my_list1 = ["b","c","a"]
del my_list1[1]
print (my_list1)
my_list1.remove("b")
print (my_list1)
```

Output

```
['b', 'a']
['a']
```

```
my_list1= [ x**2 for x in range(5)]
print (my_list1)
```

Output

```
[0, 1, 4, 9, 16]
```

```
my_list1= list( map( lambda x: x**2,range(5)) )
print (my_list1)
```

Output

```
[0, 1, 4, 9, 16]
```

```
a = "Apple"
my_list1= list(a)
print (my_list1)
i1,i2,i3,i4,i5= my_list1
print(i1,i2,i3,i4,i5)
```

Output

```
['A', 'p', 'p', 'l', 'e']
A p p l e
```

```
a = "This is my first Program"
my_list1 = a.split(' ')  # or split(delimiter)
print (my_list1)
```

Output

```
['This', 'is', 'my', 'first', 'Program']
```

Lists

```
my_list1=[1,2,1,3,4,1]
print (my_list1.count(1))
print (my_list1.index(2))
my_list1.reverse()
print (my_list1)
```

Output

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

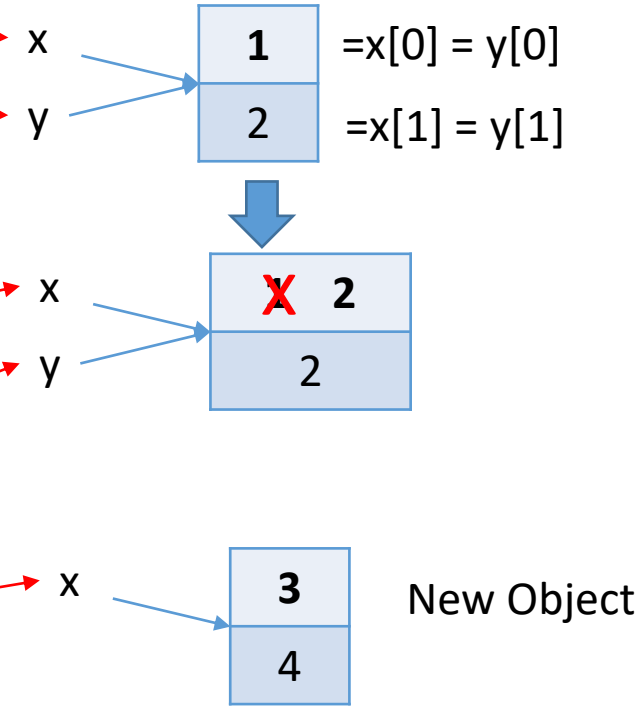
```
my_list1=[1,2,1,3,4,1]
my_list1.insert(1,"Apple")
print(my_list1)
```

Output

```
[1, 'Apple', 2, 1, 3, 4, 1]
```

Variables Assignments

```
x=[1,2]

y=x

y
Out[15]: [1, 2]




x[0]=2

y
Out[19]: [2, 2]

x=[3,4]

y
Out[21]: [2, 2]
```



List

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists.asp

Lists - Practice

- Assume the following list list1=[1,20,-1,0,1000],
  - 1- append to list1 the following list2 (list2=[23,546])
  - 2- find len, min, max, sum, sorted of the updated list1 and print the results
  - 3- Apply sort and pop and print the list

Tuples

- Tuple is a sequence of values but immutable **(Cannot be changed)**
- Tuples are **Faster** than Lists. It is used for **Fixed Data**

```
my_tuple = ( 1,20,100,5,3,20 )# or without parentheses
print (my_tuple)
```

Output

```
(1, 20, 100, 5, 3, 20)
```

```
my_list=['a','b']
t1 = ("Apple",)       # Note , is for one item
t2 = t1 + (1,2,3) + tuple(my_list)
print (t2)
print (t2[0])
print (t2[1:4])
```

Output

```
('Apple', 1, 2, 3, 'a', 'b')
Apple
(1, 2, 3)
```

```
t1=(1,['a','b','c'])
t1[1][0] = 'd'      # List is still mutable
print (t1)
del t1              # t1 is deleted
t1=1,2,3,4
print (t1)
```

Output

```
(1, ['d', 'b', 'c'])
(1, 2, 3, 4)
```

Tuples

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Sets

- A set is a collection which is **unordered** and **unindexed**
- Set is a sequence of **unique** Values
- Sets are **Faster** than Lists and **mathematical operation** can be applies.

{ }

curly brackets

```
s1={"Apple","Apple",1,1,1,2,(1,2),(1,2)}
print ( s1 )
print(list(set([1,1,2,3,4,2,2,2,2,])))
```

Output

```
{(1, 2), 1, 2, 'Apple'}
[1, 2, 3, 4]
```

```
s1={1,2,3,4,5,6}
s2={2,4,6}
print ( s1 | s2 )     # OR
print ( s1 & s2 )     # AND
print ( s1 - s2 )     # subtract
print ( s1 ^ s2 ) # in s1 or s2 but not in both
```

Output

```
{1, 2, 3, 4, 5, 6}
{2, 4, 6}
{1, 3, 5}
{1, 3, 5}
```

```
s1=set("abcdef")
s2=set("efghi")
print ( s1 | s2 )     # OR
print ( s1 & s2 )     # AND
print ( s1 - s2 )     # subtract
print ( s1 ^ s2 ) # in s1 or s2 but not in both
```

Output

```
{'i', 'g', 'h', 'a', 'e', 'b', 'c', 'd', 'f'}
{'e', 'f'}
{'a', 'c', 'd', 'b'}
{'i', 'c', 'd', 'g', 'h', 'a', 'b'}
```

Sets

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

https://www.w3schools.com/python/python_lists.asp

Dictionary

- Dictionary is a set is pairs of Data that is consist of Keys & Values
- A dictionary is a collection which is unordered, changeable and indexed

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Output

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
x = thisdict["model"]
print(x)
```

Output

```
Mustang
```

```
x = thisdict.get("model")
print(x)
```

Output

```
Mustang
```

```
thisdict["year"] = 2018
```

```
for x in thisdict:
  print(x)
```

Output

```
brand
model
year
```

```
for x in thisdict:
  print(thisdict[x])
```

```
for x in thisdict.values():
  print(x)
```

Output

```
Ford
Mustang
1964
```

Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(len(thisdict))
```

Output

```
3
```

```
for x, y in thisdict.items():
  print(x, y)
```

Output

```
brand Ford
model Mustang
year 1964
```

```
if "model" in thisdict:
  print("Yes, 'model' is one of the keys ")
```

Output

```
Yes, 'model' is one of the keys
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

Output

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

The pop() method removes the item with the specified key name:

```
thisdict.pop("model")
print(thisdict)
```

Output

```
{'brand': 'Ford', 'year': 1964, 'color': 'red'}
```

Dictionary

```
d1={
"Name":"Hussam",
"Age":47,
"Edu":"Master in Software Eng"}
print(d1)
print(d1["Name"],d1["Age"],d1["Edu"])
```

Output

```
{'Name': 'Hussam', 'Age': 47, 'Edu': 'Master in Software Eng'}
Hussam 47 Master in Software Eng
```

```
print("Name" in d1)
print("Hussam" in d1)
print("Hussam" in d1.values())
```

Output

```
True
False
True
```

```
print(d1.get("Age",50))
print(d1.get("wieght",70))
```

Output

```
47
70
```

```
my_list = list(d1)
print(my_list)
```

Output

```
['Name', 'Age', 'Edu']
```

```
my_list2=list(d1.items())#return List of Tuples
print(my_list2)
```

Output

```
[('Name', 'Hussam'), ('Age', 47), ('Edu', 'Master in Software Eng')]
```

```
print(d1.keys())
```

Output

```
dict_keys(['Name', 'Age', 'Edu'])
```

```
print(d1.values())
```

Output

```
dict_values(['Hussam', 47, 'Master in Software Eng'])
```

Dictionary

```
d2 = { 'Name': ['Hussam', 'John', 'Peter'],
     'Age': [47, 35, 70],
     'Country': ['Jordan','US','UK']}
print(d2)
print( d2["Name"][1], d2["Age"][1],d2["Country"][1])
```

Output

```
{'Name': ['Hussam', 'John', 'Peter'], 'Age':
[47, 35, 70], 'Country': ['Jordan', 'US',
'UK']}
John 35 US
```

```
d2["Age"] = [25,38,50]
print(d2["Age"])
d2["Age"][1] = 40
print(d2["Age"])
```

Output

```
[25, 38, 50]
[25, 40, 50]
```

```
del d2["Name"]
print(d2)
d2.clear()
print(d2)
```

Output

```
{'Age': [25, 40, 50], 'Country': ['Jordan',
'US', 'UK']}
{}
```

```
print( type(d2))
```

Output

```
<class 'dict'>
```

Dictionary

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Summary

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members

https://www.w3schools.com/python/python_lists.asp

Master in Software Engineering

Hussam Hourani has over 25 years of Organizations Transformation, VROs, PMO, Large Scale and Enterprise Programs Global Delivery, Leadership, Business Development and Management Consulting. His client experience is wide ranging across many sectors but focuses on Performance Enhancement, Transformation, Enterprise Program Management, Artificial Intelligence and Data Science.