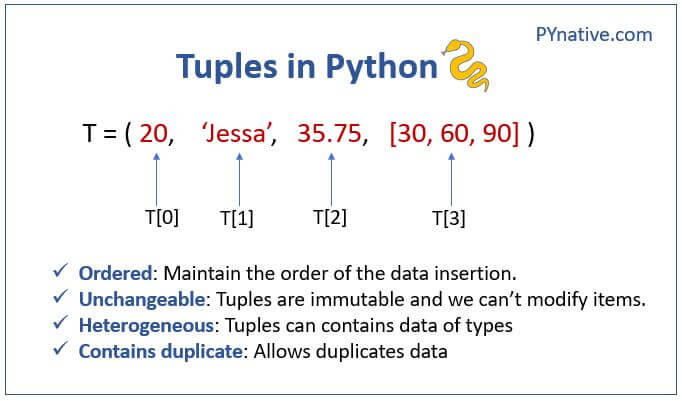
# Session 8

1. ***Tuples***
2. ***Built-in Tuple*** M***ethods***
3. ***Dictionary***
4. ***Built-in Dictionary*** M***ethods***

5. Sets

6. Built-in Set Methods

## Introduction to Tuples



***Tuples are constructed with brackets ( ) and commas separating every element in the tuple.***

***# Assign a tuple to*** *a* ***variable named my\_tup  
my\_tup = (1,2,3)  
print(my\_tup)***

***(1, 2, 3)***

***NOTE: Tuples are similar to*** lists ***in every aspect except for the fact that tuples are immutable while*** lists ***are mutable.***

***We just created a tuple of integers, but tuples can hold different object types. They can also hold duplicate elements just like lists***, For example,

***my\_tup = ('A string',23,100.232,*** ***'o')  
print(my\_tup)***

***('A string', 23, 100.232, 'o')***

***# Check len just like a list  
my\_tup = (1,2,3)  
len(my\_tup)***

***3***

***Tuple indexing and slicing also*** worksimilarly ***to list and string slicing***

***# Use indexing just like we did in lists  
my\_tup[0]***

***1***

***# Slicing just like a list  
my\_tup[-1]***

***3***

***It can't be stressed enough that tuples are immutable.*** This ***means we cannot do individual item*** assignments ***on them***

***my\_tup[0]= 'change'***

***---------------------------------------------------------------------------  
TypeError Traceback (most recent call last)  
<ipython-input-10-378d31028a15> in <module>  
----> 1 my\_tup[0]= 'change'  
  
TypeError: 'tuple' object does not support item assignment***

## Built-in Tuple Methods

***Tuples have built-in methods, but not as many as lists do. Let's look at two of them:***

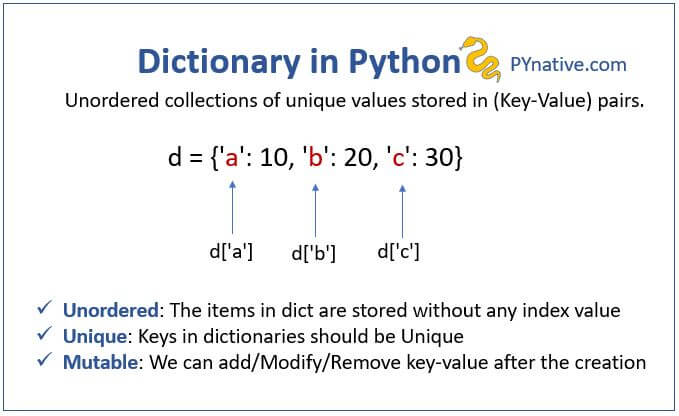
***# Use .index to enter a value and return the index  
my\_tup = ('one',2)  
my\_tup.index('one')***

***0***

***# Use .count to count the number of times a value appears  
my\_tup.count('one')***

***1***

## Introduction to Dictionary



***Dictionaries*** are ***constructed with brackets { } and commas separating every pair in the Dictionary and: separating every key and value***

***# Make a dictionary with {} and: to signify a key and a value  
my\_dict = {'key1':'value1',*** ***'key2':'value2'}***

***A key and value pair encompasses one element inside a dictionary***

***# Call values by their key  
my\_dict['key2']***

***'value2'***

***Values of the dictionary elements can be any datatype (list,* *strings,* *int,* *float*, *etc)***

***my\_dict = {'key1':123,*** ***'key2':[12,23,33],*** ***'key3':['item0',*** ***'item1','item2']}***

***The keys can only have datatype int,* *float*, *and string.***

***my\_dict = {1:123,1.1:[12,23,33],'*** ***a':['item0',*** ***'item1',*** ***'item2']}***

***We can access individual elements from the dictionary using the key name as mentioned earlier.***

***my\_dict['key3']***

***['item0', 'item1', 'item2']***

***# Can call an index on that value  
my\_dict['key3'][0]***

***'item0'***

***# Can then even call methods on that value  
my\_dict['key3'][0].upper()***

***'ITEM0'***

***We can also create keys by assignment. For instance***, ***if we started with an empty dictionary, we could continually add to it:***

***# Create a new dictionary  
d = {}***

***# Create a new key through*** *the* ***assignment  
d['animal'] = 'Dog'***

***# Can do this with any object  
d['answer'] = 42***

***#Show  
d***

***{'animal': 'Dog', 'answer': 42}***

***we can also have nested (multi-dimensional) dictionaries just like lists we can use indexing (but with key names) to grab the elements***

***# Dictionary nested inside a dictionary nested inside a dictionary  
d = {'key1':{'nestkey':{'subnestkey':'value'}}}***

***Let's see how we can grab that value:***

***# Keep calling the keys  
d['key1']['nestkey']['subnestkey']***

***'value'***

# pop()

***The pop() method removes the specified item from the dictionary. The value of the removed item is the return value of the pop() method, see*** the ***example below.***

***car = {"brand": "Ford",*** ***"model": "Mustang",*** ***"year": 1964}  
  
x = car.pop("model")  
  
print(x)***

***Mustang***

## Built-in Dictionary methods

***There are a few methods we can call a dictionary. Let's get a quick introduction to a few of them:***

***# Create a typical dictionary  
d = {'key1':1,'key2':2,'key3':3}***

***# Method to return a list of all keys   
d.keys()***

***dict\_keys(['key1', 'key2', 'key3'])***

***# Method to grab all values  
d.values()***

***dict\_values([1, 2, 3])***

***# Method to return tuples of all items (we'll learn about tuples soon)  
d.items()***

***dict\_items([('key1', 1), ('key2', 2), ('key3', 3)])***

## Iterating over dictionary

***We can use for loop to iterate over dictionary elements***

***my\_dict = {'key1':'value1',*** ***'key2':'value2',*** ***'key3':'value3',*** ***'key4':'value4'}  
  
for i in my\_dict:  
 print(i)***

***key1  
key2  
key3  
key4***

***Iterating over*** the ***dictionary only gives out the keys and not the corresponding values. We can use the methods discussed above to access the values*** of ***both the keys and values together from a dictionary***

***using* the *values() method to iterate over values***

***my\_dict = {'key1':'value1',*** ***'key2':'value2',*** ***'key3':'value3',*** ***'key4':'value4'}  
  
for i in my\_dict.values():  
 print(i)***

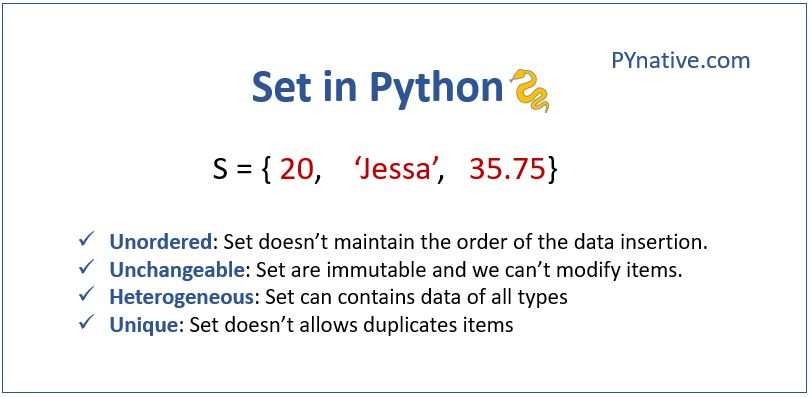
***value1  
value2  
value3  
value4***

***using* the *items() method to iterate and access both keys and values at the same time.***

***my\_dict = {'key1':'value1','key2':'value2','key3':'value3','key4':'value4'}  
  
for i in my\_dict.items():  
 print(i)***

***('key1', 'value1')  
('key2', 'value2')  
('key3', 'value3')  
('key4', 'value4')***

## Introduction to Sets



Sets are constructed with brackets { } and commas separating every element in the tuple.

In Python, a **Set is an unordered collection of data items that are unique**. In other words, Python Set is a collection of elements (Or objects) that contains no duplicate elements.

*# Assign a set to a variable named my\_set*

my\_set = (5,10,20)  
print(my\_set)

(5, 20, 10)

# create a set using {}

# set of the mixed types integer, string, and floats

set1 = {'Mark', 'Jessa', 25, 75.25}

print(set1)

# Output {25, 'Mark', 75.25, 'Jessa'}

book\_set = ("Harry Potter", "Angels and Demons", "Atlas Shrugged")

print(book\_set)

# output {'Harry Potter, 'Atlas Shrugged', 'Angels and Demons'}

print(type(book\_set))

# Output class 'set'

## Built-in Set methods

## **add( ) build-in method adds element x to the set.**

my\_set = {1, 2, 3}

my\_set.add(4)

print(my\_set) # Output: {1, 2, 3, 4}

## 

remove( ) to remove a single item from a set.

my\_set = {1, 2, 3}

my\_set.remove(2)

print(my\_set) # Output: {1, 3}

pop( ) to remove any random item from a set.

color\_set = {'red', 'orange', 'yellow', 'white', 'black'}

# Remove any random item from a set

deleted\_item = color\_set.pop()

print(deleted\_item)

color\_set = {'violet', 'indigo', 'blue', 'green', 'yellow'}

remaining\_colors = {'indigo', 'orange', ‘blue’, 'red'}

union() to the union of two sets using the OR operator

# union using union() method

vibgyor\_colors = color\_set.union(remaining\_colors)

print(vibgyor\_colors)

# Output {'indigo', 'blue', 'violet', 'yellow', 'red', 'orange', 'green'}

## HOMEWORK

### 1. Print the following pattern using nested for loop

***Size: 7 x 21   
---------.|.---------  
------.|..|..|.------  
---.|..|..|..|..|.---  
-------WELCOME-------  
---.|..|..|..|..|.---  
------.|..|..|.------  
---------.|.---------***

#### (A simple Google search will give you the solution. But where is the fun in it!!!!)

## HOMEWORK SOLUTION

***#TASK 1:  
#TASK 2:  
rows = int(input("enter*** a ***number of rows: "))  
  
for x in range(rows):  
 for i in range(x,*** ***rows):  
 print("-",*** ***end = '')  
 print()***

***enter*** a ***number of rows: 5  
-----  
----  
---  
--  
-***

1. ***Draw pattern***

***P Py Pyt Pyth Pytho Python***

***#TASK 3: Reverse the tuple(Hint:*** ***use slicing)***

***#TASK 4: Count the number of occurrences of item 50 from the given tuple. tuple1 = (50, 10, 60, 70, 50)***

***#TASK 5:  
Access value 20 from the given tuple.  
tuple1 = ("Orange", [10, 20, 30], (5, 15, 25)) (Hint: use indexing)***

***#TASK 6: Create a tuple with*** a ***single item 80***

***#TASK 7:  
Write a Python program to iterate over dictionaries using loops for the given dictionary.  
d = {'x': 10, 'y': 20, 'z': 30}***

***#TASK 8:  
Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the   
values are*** squares ***of keys.***

***#TASK 9: Write a Python program to check*** whether ***a dictionary is empty or not.(Hint: use Boolean)***

***#TASK 10: Create a dictionary by extracting the keys from the given dictionary. sampleDict = {"name": "Kelly",*** ***"age":25,*** ***"salary": 8000,*** ***"city": "New*** York***" }***

***#TASK 11: Write a program to rename a key city to a location in the following dictionary. sample\_dict = {"name": "Kelly",*** ***"age":25,*** ***"salary": 8000,*** ***"city": "New*** York***"}***