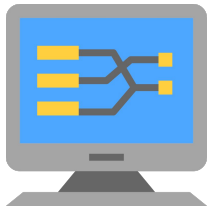
Dictionaries (Hash Maps)

Python dictionaries are unordered collection of pairs in the form of **key:value.**

e.g.
```
L=[ 2,4,5,6,10 ]
print(L[2])

D = { 'dog' : 'friendly' , 'lion' : 'wild' , 'parrot' : 'funny' }
print( D['dog'] )
```



Dictionaries

- It’s an unordered set of **key:value** pairs.
- Dictionaries are mutable.
- Unlike a string, list & tuple. A dictionary is not a sequence, it’s unordered set of elements.
- All keys must be unique.
- We can change the value of any key :

  D[<key>] = <new value>

Playing with Dictionaries

- Walking in a Dictionary
  To traverse in a dictionary we can use the **for loop**.

```
for <variable> in <dictionary> :
        print(<variable>)
```

```
for i in d :
        print( i, ':', d[i] )
```

D = { 'dog' : 'friendly' , 'lion' : 'wild' , 'parrot' : 'funny' }

Playing with Dictionaries

- Adding an element
  D['monkey'] = 'Human-Like'
- Updating an element
  D['parrot'] ='Sometimes-funny' ✅
  D['bat'] ='Not-for-eating' ❌

Playing with Dictionaries

- Deleting an element
  del <dictionary>[<key>] del d['lion']
  <dictionary>.pop(<key>) d.pop('lion')
  Pop method also returns the deleted element.
  print(d.pop('lion'))
- Checking existence of a key
  'lion' in d --- true
  'tiger' in d --- false

Dictionaries Functions

- The len( ) method
  len(<dictionary>) --- gives the count of pairs
- The clear( ) method
  <dictionary>.clear( ) --- clears the whole dictionary.
- The get( ) method
  print(d.get('lion')) --- print(d['lion'])
- The keys( ) method
  print(d.keys( )) --- displays all the keys

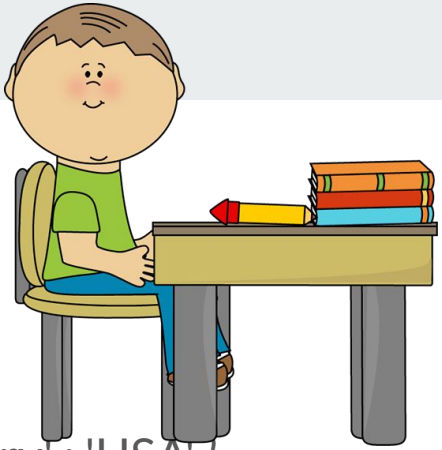
Dictionaries Functions

- The items( ) method
  This method returns all the pairs as a sequence of (key,value) tuples.
  (in no particular order).
  `L=d.items( )`
  `print(L)` --- [('dog' , 'friendly'),('lion' , 'wild'),('parrot' , 'funny')]
- The values( ) method
  `<dictionary>.values( )` --- returns all the values.

Dictionaries Functions

- The update( ) method
  <dictionary>.update(<other-dictionary>)
  <other-dictionary> will over-ride in <dictionary>.
  - New Keys will be added.
  - Existing key will be updated with new values.

  e.g. d = { 'dog' : 'friendly' , 'lion' : 'wild' , 'parrot' : 'funny' }
  d2= { 'dog' : pet' , 'lion' : 'wild' , tiger' : 'dangerous' }
  d.update(d2)
  print(d)
  { 'dog' : pet' , 'lion' : 'wild' , 'parrot' : 'funny', 'tiger':'dangerous' }



Practice Time

Q1.
```
R={ 'Name' : 'Python', 'Age' : '20', 'Addr' : 'NJ' , 'Country' : 'USA' }
id1=id(R)
R2={ 'Name' : 'Python', 'Age' : '20', 'Addr' : 'NJ' , 'Country' : 'USA' }
id2=id(R2)
print(id1==id2)
```
(id( ) gives the address of the object)

Q2.
```
D = { }
D[ (1,2,4) ] =8
D[ (4,2,1) ] =10
D[ (1,2) ] =12
sum = 0
```

```
for k in D:
   sum+=D[k]
print(sum)
print(D)
```

Practice Time

1. W.a.p. to verify whether the phone number is in the given format or not
   (ex. 017-555-1212 is the correct format)

2. Write a program that prints the longest word in a list.
3. Write a python program that creates a list storing first 9 terms of fibonacci series.
   1, 1, 2, 3 ,5 ,8 ….

4. **W.a.p. to make a list of all integers less than hundred which are multiples of 3 or 5. --- H.W.**

Practice Time

1. Write a function addDict(dict1, dict2) which computes the union of two dictionaries. If same key appears in both, pick either one.

2. Create a dictionary whose keys are *month names* and whose values are *number of days* in the corresponding months.
   a. Ask user to enter month name and number of days.
   b. Print out all of the keys in alphabetical order.
   c. Print out all of the months with 31 days.
   d. Print out the (key-value) pairs sorted by keys.
   e. Print out the (key-value) pairs sorted by values.

Typical

THANK YOU FOR WATCHING!

Milte hain next video me, BYEE!!!!