Dictionaries

- It is a collection of hetrogenoues data set like numaric,alpla,char,str,spechar..etc
- Data in the form of key and value pair
- it contains unorder data
- Every is unique data
- It is mutable
- short form is dict
- It is also in the reprasentation of like {} but separated by colon(:) key and value

In [2]:
```
d={"key1":12122,"key2":7521,"key3":753713}
d
```

Out[2]:
```
{'key1': 12122, 'key2': 7521, 'key3': 753713}
```

In [3]:
```
print(dir(dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'cle
ar', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefaul
t', 'update', 'values']
```

In [5]:
```
d.clear()
d
```

Out[5]:
```
{}
```

In [6]:
```
nd={"Cbit":2008,"Apssdc":2014,"Python":1991}
nd
```

Out[6]:
```
{'Cbit': 2008, 'Apssdc': 2014, 'Python': 1991}
```

In [7]:
```
nd.keys()#for all keys form dict..
```

Out[7]:
```
dict_keys(['Cbit', 'Apssdc', 'Python'])
```

In [8]:
```
nd.values()#for all values form dict..
```

Out[8]:
```
dict_values([2008, 2014, 1991])
```

In [9]:
```
nd1={"Kadapa":"Cbit","Vizag":"Gitam","Vijayawada":"Velagapudi Ramakrishna SE
nd1
```

Out[9]:
```
{'Kadapa': 'Cbit',
 'Vizag': 'Gitam',
 'Vijayawada': 'Velagapudi Ramakrishna SEC'}
```

In [10]:
```
1 nd.update(nd1)
```

In [11]:
```
1 nd
```

Out[11]:
```
{'Cbit': 2008,
 'Apssdc': 2014,
 'Python': 1991,
 'Kadapa': 'Cbit',
 'Vizag': 'Gitam',
 'Vijayawada': 'Velagapudi Ramakrishna SEC'}
```

In [12]:
```
1 nd.popitem()
```

Out[12]:
```
('Vijayawada', 'Velagapudi Ramakrishna SEC')
```

In [13]:
```
1 nd.pop("Vizag")
```

Out[13]:
```
'Gitam'
```

In [14]:
```
1 nd
```

Out[14]:
```
{'Cbit': 2008, 'Apssdc': 2014, 'Python': 1991, 'Kadapa': 'Cbit'}
```

In [17]:
```
1 nd["Python"]=1989#update paticular value
```

In [18]:
```
1 nd
```

Out[18]:
```
{'Cbit': 2008, 'Apssdc': 2014, 'Python': 1989, 'Kadapa': 'Cbit'}
```

In [ ]:
```
1 #Task
2     #Funtion to create one contact app using dictionary
3
4
```

In [19]:
```
1 contacts={}
```

In [56]:
```
1  #Add contact to dict
2  def addContact(name,number):
3      if name not in contacts:
4          contacts[name]=number
5          print("Contact added successfully..")
6      else:
7          print("Contact already exist : ",name)
8  name=input("Enter person name :")
9  number=int(input("Enter person number :"))
10 addContact(name,number)
11
```

```
Enter person name :ts
Enter person number :53
Contact added successfully..
```

In [55]:
```
contacts
```

Out[55]:
```
{'muni': 565, 'vijay': 12, 'cr': 6735}
```

In [31]:
```
#update the contact
def updateContact(name):
    if name in contacts:
        number=int(input("Enter the contact for update..."))
        contacts[name]=number
        print("Successfully updated",name ,"in",number)
    else:
        print("Not Exists for update :",name)
updateContact("vijay")

```

```
Enter the contact for update...s
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-37f40fcef43f> in <module>
      7     else:
      8         print("Not Exists for update :",name)
----> 9 updateContact("vijay")

<ipython-input-31-37f40fcef43f> in updateContact(name)
      2 def updateContact(name):
      3     if name in contacts:
----> 4         number=int(input("Enter the contact for update..."))
      5         contacts[name]=number
      6         print("Successfully updated",name ,"in",number)

ValueError: invalid literal for int() with base 10: 's'
```

In [32]:
```
#update the contact
def deletecontact(name):
    if name in contacts:
        contacts[name]=number
        contacts.pop(name)
        print("Successfully deleted",name ,"in",number)
    else:
        print("contact not Exists for delete :",name)
deletecontact("vijay")

```

```
Successfully deleted vijay in 676444
```

In [33]:
```
contacts
```

Out[33]:
```
{'muni': 12345}
```

In [49]:
```
#Display contacts
#       1.ap-->123
#       2.ts-->6565
#       3.tn-->7676
#       4.muni-->12345

def dcontacts(d):
    c=1
    if len(contacts)<0:
        for name in contacts:
            if name in contacts:
                print(c,".",name,"---->",number)
                c=c+1
    else:
        print("contacts is empty")

dcontacts(contacts)
```

```
contacts is empty
```

In [43]:
```
contacts.clear()

```

In [58]:
```
contacts
```

Out[58]:
```
{'muni': 565, 'vijay': 12, 'cr': 6735, 'ts': 53}
```

In [62]:
```
def dcontacts(d):
    c=1
    if len(contacts)>0:
        for n,nm in contacts.items():
            c=c+1
            print(c,".",n,"---->",nm)

    else:
        print("contacts is empty")
#             if name in contacts:
dcontacts(contacts)
```

```
2 . muni ----> 565
3 . vijay ----> 12
4 . cr ----> 6735
5 . ts ----> 53
```

In [81]:
```
# Generation of RollNumbers
Roll_number=[]
def Generate_RollNumber(lb,ub):
    for i in range(lb,ub+1):
        if(i==57 or i==80 or i==75 or i==84):
            continue
        else:
            Roll_number.append("172p1a05"+str(i))
Generate_RollNumber(51,94)
print(Roll_number)

```

```
['172p1a0551', '172p1a0552', '172p1a0553', '172p1a0554', '172p1a0555', '172p1a0
556', '172p1a0558', '172p1a0559', '172p1a0560', '172p1a0561', '172p1a0562', '17
2p1a0563', '172p1a0564', '172p1a0565', '172p1a0566', '172p1a0567', '172p1a056
8', '172p1a0569', '172p1a0570', '172p1a0571', '172p1a0572', '172p1a0573', '172p
1a0574', '172p1a0576', '172p1a0577', '172p1a0578', '172p1a0579', '172p1a0581',
'172p1a0582', '172p1a0583', '172p1a0585', '172p1a0586', '172p1a0587', '172p1a05
88', '172p1a0589', '172p1a0590', '172p1a0591', '172p1a0592', '172p1a0593', '172
p1a0594']
```

In [86]:
```
# Generate Marks
marks_list=[]
import random
def Generate_marks(lb,ub):
    for i in range(lb,ub+1):
        v=random.randint(1,100)
        marks_list.append(v)
Generate_marks(51,94)
print(marks_list)
```

```
[87, 12, 97, 96, 10, 90, 25, 6, 55, 27, 75, 61, 97, 69, 45, 51, 27, 62, 74, 26,
58, 2, 39, 51, 21, 86, 55, 27, 73, 10, 42, 46, 32, 79, 27, 56, 16, 22, 98, 37,
63, 67, 21, 44]
```

In [83]:
```
print(Roll_number)
print(marks_list)
```

```
['172p1a0551', '172p1a0552', '172p1a0553', '172p1a0554', '172p1a0555', '172p1a0
556', '172p1a0558', '172p1a0559', '172p1a0560', '172p1a0561', '172p1a0562', '17
2p1a0563', '172p1a0564', '172p1a0565', '172p1a0566', '172p1a0567', '172p1a056
8', '172p1a0569', '172p1a0570', '172p1a0571', '172p1a0572', '172p1a0573', '172p
1a0574', '172p1a0576', '172p1a0577', '172p1a0578', '172p1a0579', '172p1a0581',
'172p1a0582', '172p1a0583', '172p1a0585', '172p1a0586', '172p1a0587', '172p1a05
88', '172p1a0589', '172p1a0590', '172p1a0591', '172p1a0592', '172p1a0593', '172
p1a0594']
[67, 75, 62, 87, 96, 99, 90, 92, 68, 62, 54, 76, 88, 96, 87, 80, 94, 50, 71, 8
1, 68, 53, 98, 88, 72, 95, 92, 97, 77, 77, 56, 74, 80, 84, 67, 51, 55, 98, 64,
97, 85, 70, 78, 94]
```

In [88]:
```
# Storing Roll Numbers and Marks in to Dict
student_data={}
for i in range(len(Roll_number)):
    student_data[Roll_number[i]]=marks_list[i]
print(student_data)

```

```
{'172p1a0551': 87, '172p1a0552': 12, '172p1a0553': 97, '172p1a0554': 96, '172p1
a0555': 10, '172p1a0556': 90, '172p1a0558': 25, '172p1a0559': 6, '172p1a0560':
55, '172p1a0561': 27, '172p1a0562': 75, '172p1a0563': 61, '172p1a0564': 97, '17
2p1a0565': 69, '172p1a0566': 45, '172p1a0567': 51, '172p1a0568': 27, '172p1a056
9': 62, '172p1a0570': 74, '172p1a0571': 26, '172p1a0572': 58, '172p1a0573': 2,
'172p1a0574': 39, '172p1a0576': 51, '172p1a0577': 21, '172p1a0578': 86, '172p1a
0579': 55, '172p1a0581': 27, '172p1a0582': 73, '172p1a0583': 10, '172p1a0585':
42, '172p1a0586': 46, '172p1a0587': 32, '172p1a0588': 79, '172p1a0589': 27, '17
2p1a0590': 56, '172p1a0591': 16, '172p1a0592': 22, '172p1a0593': 98, '172p1a059
4': 37}
```

In [95]:
```
# To find Failed Count As well as Passed Count
student_data.keys()
student_data.values()
student_data.items()
fail_count=0
pass_count=0
for value in student_data.values():
    if(value>=35):
        pass_count+=1
    else:
        fail_count+=1
print("Total Students=",fail_count+pass_count)
print("Failed Count=",fail_count)
print("Passed Count=",pass_count)
```

```
Total Students= 40
Failed Count= 15
Passed Count= 25
```

In [97]:
```
# Swapping Roll numbers to marks because of to find fail and pass data
ana_studetn_data={}
for i in range(len(Roll_number)):
    ana_studetn_data[marks_list[i]]=Roll_number[i]
print(ana_studetn_data)

```

```
{87: '172p1a0551', 12: '172p1a0552', 97: '172p1a0564', 96: '172p1a0554', 10: '1
72p1a0583', 90: '172p1a0556', 25: '172p1a0558', 6: '172p1a0559', 55: '172p1a057
9', 27: '172p1a0589', 75: '172p1a0562', 61: '172p1a0563', 69: '172p1a0565', 45:
'172p1a0566', 51: '172p1a0576', 62: '172p1a0569', 74: '172p1a0570', 26: '172p1a
0571', 58: '172p1a0572', 2: '172p1a0573', 39: '172p1a0574', 21: '172p1a0577', 8
6: '172p1a0578', 73: '172p1a0582', 42: '172p1a0585', 46: '172p1a0586', 32: '172
p1a0587', 79: '172p1a0588', 56: '172p1a0590', 16: '172p1a0591', 22: '172p1a059
2', 98: '172p1a0593', 37: '172p1a0594'}
```

In [102]:
```
# to separate failed data and passed data in to two dictinaries
fail_list={}
pass_list={}
print("Passed Data:")
for mark,roll in ana_studetn_data.items():
    if(mark>=35):
        print(ana_studetn_data[mark],"=",mark)
        pass_list[ana_studetn_data[mark]]=mark
print("Failed Data:")
for mark,roll in ana_studetn_data.items():
    if(mark<35):
        print(ana_studetn_data[mark],"=",mark)
        fail_list[ana_studetn_data[mark]]=mark


```

```
Passed Data:
172p1a0551 = 87
172p1a0564 = 97
172p1a0554 = 96
172p1a0556 = 90
172p1a0579 = 55
172p1a0562 = 75
172p1a0563 = 61
172p1a0565 = 69
172p1a0566 = 45
172p1a0576 = 51
172p1a0569 = 62
172p1a0570 = 74
172p1a0572 = 58
172p1a0574 = 39
172p1a0578 = 86
172p1a0582 = 73
172p1a0585 = 42
172p1a0586 = 46
172p1a0588 = 79
172p1a0590 = 56
172p1a0593 = 98
172p1a0594 = 37
Failed Data:
172p1a0552 = 12
172p1a0583 = 10
172p1a0558 = 25
172p1a0559 = 6
172p1a0589 = 27
172p1a0571 = 26
172p1a0573 = 2
172p1a0577 = 21
172p1a0587 = 32
172p1a0591 = 16
172p1a0592 = 22
```

In [104]:
```
print("Passed Data is")
for roll,mark in pass_list.items():
    print(roll,"-->",mark)
```

```
Passed Data is
172p1a0551 --> 87
172p1a0564 --> 97
172p1a0554 --> 96
172p1a0556 --> 90
172p1a0579 --> 55
172p1a0562 --> 75
172p1a0563 --> 61
172p1a0565 --> 69
172p1a0566 --> 45
172p1a0576 --> 51
172p1a0569 --> 62
172p1a0570 --> 74
172p1a0572 --> 58
172p1a0574 --> 39
172p1a0578 --> 86
172p1a0582 --> 73
172p1a0585 --> 42
172p1a0586 --> 46
172p1a0588 --> 79
172p1a0590 --> 56
172p1a0593 --> 98
172p1a0594 --> 37
```

In [105]:
```
print("Failed Data is ")
for roll,mark in fail_list.items():
    print(roll,"-->",mark)
```

```
Failed Data is
172p1a0552 --> 12
172p1a0583 --> 10
172p1a0558 --> 25
172p1a0559 --> 6
172p1a0589 --> 27
172p1a0571 --> 26
172p1a0573 --> 2
172p1a0577 --> 21
172p1a0587 --> 32
172p1a0591 --> 16
172p1a0592 --> 22
```

In [ ]:
```
#Task -
#Create Login and signup page using dict..
    #explanation:
        #LoginCredentials{"user1":["password","mobile","email"]
        #                  "user2":"password2"
        #                   "user3":"password3"}
    #store the   LoginCredentials to dict
    # if check the login working or not


```

Regular Expressions

```
* It ia programming lan..
* It matchs the patterns by using symbols
* helpful for reduce the no.of lines in the coding
* it is 100% accurate
```

Regular Expression

- A Regular Expression (RegEx) is a sequence of characters that match a pattern. It is a symbolic representation. To use regular expressions, we have to import regular expressions(import re) while writing a program.
  - Character
    - Use case _ []-->Represent a character
      - ^ -->Matches the beginning
      - $ -->Matches the ending
      - -->Number of occurrences(zero also)
      - -->Number of occurrences(from 1)
      - . -->To match a character
      - ? -->For zero or one occurrences
      - | -->OR
      - {} -->Range set
      - () -->For enclosing of regular expression

In [106]:
```
# import re
import re

print(dir(re))

```

```
['A', 'ASCII', 'DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTI
LINE', 'Match', 'Pattern', 'RegexFlag', 'S', 'Scanner', 'T', 'TEMPLATE', 'U',
'UNICODE', 'VERBOSE', 'X', '_MAXCACHE', '__all__', '__builtins__', '__cached_
_', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
'__version__', '_cache', '_compile', '_compile_repl', '_expand', '_locale', '_p
ickle', '_special_chars_map', '_subx', 'compile', 'copyreg', 'enum', 'error',
'escape', 'findall', 'finditer', 'fullmatch', 'functools', 'match', 'purge', 's
earch', 'split', 'sre_compile', 'sre_parse', 'sub', 'subn', 'template']
```

In [108]:
```
#match

re.match("c","apssdc")#It is knothng return

```

In [109]:
```
re.match("a","apssdc")#It only first letter given str
```

Out[109]:
```
<re.Match object; span=(0, 1), match='a'>
```

In [110]:
```
re.match("p","apssdc")
```

In [111]:
```
#search
re.search("c","apssdc")#return only one chr any position
```

Out[111]:
```
<re.Match object; span=(5, 6), match='c'>
```

In [112]:
```
re.search("s","apssdc")
```

Out[112]:
```
<re.Match object; span=(2, 3), match='s'>
```

In [113]:
```
#findall
re.findall("s","apssdc")
```

Out[113]:
```
['s', 's']
```

In [ ]:
```
#All char
    #[a-z]
#All num
    #[0-9]
#Multilpy of 5
    #[0-9]*[5]
#even numbers
    #^[0-9][2346]$
#Odd number two digit
    #^[0-9][1357]$
#Four digit num stars with 4 and ends with 8
    #^[4][0-9]{2}[8]$
#Word matching
    #^(dhoni)$ or ^[d][h][o][n][i]$
#Mobile number validation
    #case-1:it is starting with 6 or 7 or 8 or 9 and ten digits
        # ^[6789][0-9]{9}$
    #case-2:it is starting with 6 or 7 or 8 or 9 and ten digits also startin
        #ex.09876543212
        #^[0][6-9][0-9]{9}$
    #case-2:it is starting with 6 or 7 or 8 or 9 and ten digits also startin
    #Final mobile number Validation
    #^[6789][0-9]{9}$|^[0][6-9][0-9]{9}$|^[+][9][1][6-9][0-9]{9}$
#Email validation
    #ex:example@domin.com
        #1.username: len->8 to 18
            #rulz:doen't statring number,spl symbols
        #2.domine name:len->3 to 7
        #3.extenstion:len ->2 to 4
        #Final:^[a-zA-Z][a-z0-9._]{7,18}[@][a-z]{3,7}[.][a-z]{2,3}$

```

In [116]:
```
import re

pattern="^[a-zA-Z][a-z0-9._]{7,18}[@][a-z]{3,7}[.][a-z]{2,3}$"

def isValidEmail(email_id):
    if (re.match(pattern,email_id)):
        print("Valid ",email_id)
    else:
        print("Invalid :",email_id)
email_id=input("Enter the Email id :")
isValidEmail(email_id)


```

```
Enter the Email id :pguruneelima123@gmail.com
Valid  pguruneelima123@gmail.com
```

Packages and modules

- Set of funtions called modules
- Set of modules called packages
- It just know as one python(.py)
- we will import that packages by using import keyword

In [117]:
```
#Two types
#     1.predefined
#     2.user defined
import math

print(dir(math))
#ceil
#floor
a=5
b=2
a/b

```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acos
h', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cos
h', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floo
r', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfini
te', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'mod
f', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'tau', 'trunc']
```

In [124]:
```
a=10
b=2
math.floor(a/b)
```

Out[124]: 3

In [125]:
```
a=5
b=2
math.ceil(a/b)
```

Out[125]:
```
3
```

In [128]:
```
#User defiend packages
def isPrime(n):
    if n<2:
        return False
    for i in range(2,(n//2)+1):
        if n%i==0:
            return False
    return True
isPrime(10)
```

Out[128]:
```
False
```

In [129]:
```
import mypackage as p

p.isPrime(23)


```

Out[129]:
```
True
```

In [1]:
```
import Data.mypck as dm
```

In [2]:
```
dm.sum_naturalnumbers(10)
```

```
55
```

In [ ]:
```

```