In [6]:

```
# dictionary
# sets
# functions
```

dictionary

1. **it is used to store the collection of data or the information**
2. **these unodered and changable**
3. **combination of keys and values seperated by ":"**
4. **{keys : values}**

In [8]:

```
d={}
d1=dict()
print(type(d))
print(type(d1))
```

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

In [18]:

```
d={1:28,2:"harsha",3:"joel"}
print (d)
print(d[1])
print(d[2])
print(d[3])
```

```
{1: 28, 2: 'harsha', 3: 'joel'}
28
harsha
joel
```

In [17]:

```
l=[1,2,3,4,28,'harsha']
print (l[-2])#index position
```

```
28
```

In [21]:

```
#zip is used to change lists and tuples to dictionary
l1=[1,2,3,4,5]
l2=[10,20,30,40,50]
d=dict(zip(l1,l2))
print (d)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
```

In [24]:

```
l=[1,2,3,4,5,5]#will not take repeated values
t=('a','b','c','d','e',7)
d=dict(zip(l,t))
print(d)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 7}
```

In [25]:

```
print (dir (dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq
```

```
', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__in
it__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__
', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__setattr__', '__setitem__
', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'item
s', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [32]:

```
d={'name':'harsha','branch':'cse','rollno':42,'clg':'kits'}
print(d.keys())
print (d.values())
print (d.items())
print(d.get('branch'))
print (d['branch'])
```

```
dict_keys(['name', 'branch', 'rollno', 'clg'])
dict_values(['harsha', 'cse', 42, 'kits'])
dict_items([('name', 'harsha'), ('branch', 'cse'), ('rollno', 42), ('clg', 'kits')])
cse
cse
```

In [33]:

```
d={'name':'harsha','branch':'cse','rollno':42,'clg':'kits'}
d['age']=19  #adding
print(d)
```

```
{'name': 'harsha', 'branch': 'cse', 'rollno': 42, 'clg': 'kits', 'age': 19}
```

In [36]:

```
d={'name':'harsha','branch':'cse','rollno':42}
#delete
print(d.pop('rollno'))
print(d)
```

```
42
{'name': 'harsha', 'branch': 'cse'}
```

In [38]:

```
d={'name':'harsha','branch':'cse','rollno':42}
d.popitem()
print(d)
#deletes last item
```

```
{'name': 'harsha', 'branch': 'cse'}
```

In [40]:

```
d={'name':'harsha','branch':'cse','rollno':42}
#edits dictionary
(d.update({'branch':'ece'}))#key
print (d)
```

```
{'name': 'harsha', 'branch': 'ece', 'rollno': 42}
```

In [43]:

```
d={'name':'harsha','branch':'cse','rollno':42}
d.update({'cse':'it'})
print(d)
#not updated
#created new one
#bcoz key not used
```

```
{'name': 'harsha', 'branch': 'cse', 'rollno': 42, 'cse': 'it'}
```

In [47]:

```
d={'name':'harsha','branch':'cse','rollno':42}
```

```
d.setdefault('age',[19,20])
print(d)
d.setdefault('age',[20,21])
print(d)
#first values are fixed
#update can b used
```

```
{'name': 'harsha', 'branch': 'cse', 'rollno': 42, 'age': [19, 20]}
{'name': 'harsha', 'branch': 'cse', 'rollno': 42, 'age': [19, 20]}
```

In [51]:

```
a={}
print(a.fromkeys(('a','b','c'),[1,2,3]))
#if values not given o/p is none
```

```
{'a': [1, 2, 3], 'b': [1, 2, 3], 'c': [1, 2, 3]}
```

In [130]:

```
#copy
a={'a':2,'b':3}
b=a.copy()
print(b)
print(a)
```

```
{'a': 2, 'b': 3}
{'a': 2, 'b': 3}
```

In [60]:

```
d1={1:10,2:20}
d2={3:30,4:40}
d3={5:50,6:60}
d4={}
#{1:10,2:20,3:30,4:40,5:50,6:60}
for d in (a,d2,d3):
    d.update(e)
print(d)
```

```
{5: 50, 6: 60, 1: 50, 2: 60}
```

In [62]:

```
#engineering
#{'e':(3),'n':(3).....}
a=input("enter a string:")
b={}
for i in a:
    s=a.count(i)
    b[i]=s #{'e'}
print (b)
```

```
enter a string:engineering
{'e': 3, 'n': 3, 'g': 2, 'i': 2, 'r': 1}
```

In [66]:

```
d={'name':'harsha','branch':'cse','rollno':42}
del d['branch']
print(d)
```

```
{'name': 'harsha', 'rollno': 42}
```

In [72]:

```
stdt={1:['harsha',42],2:['joel',49],3:['ravi',47]}
#4th value should be added
#1st value should be deleted#
#2,3,4 keys & their o/p S should be printed
stdt.pop(1)#del stdt[1]
stdt[4]=['gopi',61]#update also can used
```

```
print(stdt)
for k,v in stdt.items():
    print(k,":",v)
```

```
{2: ['joel', 49], 3: ['ravi', 47], 4: ['gopi', 61]}
2 : ['joel', 49]
3 : ['ravi', 47]
4 : ['gopi', 61]
```

sets

sets

- **collection of data**
- **it is unordered and unindexed**
- **{} is used**

In [74]:

```
s={}
print(type (s))
s1=set(s)
print (type(s1))
```

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

In [78]:

```
s={1,2,3,4,5,6,7}
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7}
```

In [80]:

```
s={1}
s.add(2)
print(s)
```

```
{1, 2}
```

In [83]:

```
s={1,3,5}
s1=s.copy()
print(s1)
print(s)
```

```
{1, 3, 5}
{1, 3, 5}
```

In [86]:

```
s1={1,6,7,8,2}
s2={6,7,8,9,0}
print(s1.difference(s2))#s1-s2
print(s2.difference(s1))#s2-s1
```

```
{1, 2}
{0, 9}
```

In [90]:

```
s1={1,6,7,8,2}
s2={6,7,8,9,0}
s2.discard(8)
print(s2)
```

```
{0, 6, 7, 9}
```

In [94]:

```
s1={1,6,7,8,2}
s2={6,7,8,9,0}
print(s1.intersection(s2))#repeated
print(s1.union(s2))#unique
```

```
{8, 6, 7}
{0, 1, 2, 6, 7, 8, 9}
```

Functions

Functions

- **functions are defined as block of reusable code**
- **converts large programm into building blocks**
- **2 types**
  1. **pre-defined or builtin**
  2. **user defined**
- **1.(int,max,min,sum,len,ord,chr,float,map,filter)**
- **2.(using def keyword)**

In [96]:

```
l=[1,2,3,4,5]
print(sum(l))
print(len(l))
```

```
15
5
```

In [98]:

```
a=10
b=(-29)
c=5.67
print(abs(a))
print(abs(b))
print(abs(c))
#absolute values
# removes "-ve"
```

```
10
29
5.67
```

syntax:

- **function definition**

1. **def functionname(arguments):**
   - **body of function**
2. **function calling**
3. **functionname()**

In [123]:

```
def hello():
    print ("hello world")
    return
hello()
```

```
hello world
```

hello world

In [108]:

```
def add():
    a=10
    b=20
    c=a+b
    return c
print (add())
```

30

In [110]:

```
#without arguments without return values
def addition():
    a=10
    b=3
    c=a+b
    print (c)
addition()
```

13

In [114]:

```
#with arguments without return value
def multiplication(a=12,b=2):
    c=(a*b)
    print (c)
multiplication()
```

24

In [116]:

```
def multiplication(a,b):
    c=(a*b)
    print (c)
a=int(input())
b=int(input())
multiplication(a,b)#a,b must
```

2
4
8

In [128]:

```
#without arguments with return values
def power():
    n=2
    m=3
    a=(n**m)
    #print (a)
    return print(a)
power()
```

8

In [129]:

```
#with arguments with return value
#variable len argument
def printnames(*names):
    # print(names)
    return print(names)
printnames('a','b','c')
```

('a', 'b', 'c')

Assignment:

- **functions to print multiplication table with in a range (1 to 10)?**
- **function to print cubes of even numbers?**
- **function to return the average of cube of even nuumbers in a given range ?**