Today Topics

- Data Structures
- DA Modules

Data Structue

- structuring of data
- way of organizing the data in a particular format
- 4 data structures
- 1.tuple
- 2.list
- 3.set
- 4.dictionary

tuple

- one of the data structures in python allows the user/programmer to store heterogeneous data items
  - it can store different type pf data at a time
- it is represented by(paranthesis)
- tuple() is the pre-defined function
- it is immutable measn cannot be modified further after initialization
- 2 methods
  - count
  - index

In [22]:
```
tp=(1,2,'word',90.45,'python','(4+5j)',True,bin(6))
print(tp)
```
```
(1, 2, 'word', 90.45, 'python', '(4+5j)', True, '0b110')
```

In [8]:
```
for item in tp:
    print(item)
```
```
1
2
word
90.45
python
(4+5j)
```

In [9]:
```
# using index
tp[3]# 4th element present in 3rd index
```
Out[9]:
```
90.45
```

```
In [23]: tp[-1]# last element

Out[23]: '0b110'

In [11]: tp[-2] # last but one

Out[11]: 'python'

In [12]: # slice means extarcting same part of iterable
         # using index

In [24]: tp[::]

Out[24]: (1, 2, 'word', 90.45, 'python', '(4+5j)', True, '0b110')

In [15]: tp[::-1] # reversed iterable

Out[15]: ('(4+5j)', 'python', 90.45, 'word', 2, 1)

In [16]: tp[::2] # alternate valuees

Out[16]: (1, 'word', 'python')

In [18]: tp[::-2] # alternate items in the reverse order

Out[18]: ('(4+5j)', 90.45, 2)

In [19]: tp[2:5] # upper bound is exclusive

Out[19]: ('word', 90.45, 'python')

In [20]: tp[:4]# starts from first by default

Out[20]: (1, 2, 'word', 90.45)

In [21]: tp[3:] # up to the end

Out[21]: (90.45, 'python', '(4+5j)')

In [25]: bin(9) # binary format of value also stored in str format

Out[25]: '0b1001'

In [ ]: ## Method on tuple
        count() # frequency of item
        # no .of occurence of data item
```

In [28]:
```
tp2=tuple(input().split())
tp2
```

```
python workshop 235 908 college @@#%*()
```

Out[28]:
```
('python', 'workshop', '235', '908', 'college', '@@#%*()')
```

In [29]:
```
# print the values in tuple tp2
for item in tp2:
    if item.isnumeric():
        print(item,end=" ")
```

```
235 908
```

In [35]:
```
t=(3,4,5,'word',90,34,'workshop','srkit',9.3,7,3,4,3)
# to print words/str
for item in t:
    if type(item)==str:
        print(item,end=" ")
```

```
word workshop srkit
```

In [36]:
```
for item in t:
    if type(item)==int:
        print(item,end=" ")
```

```
3 4 5 90 34 7 3 4 3
```

In [37]:
```
# find the frequency of value of 3
count=0
for val in t:
    if val==3:
        count+=1
print(count)
```

```
3
```

In [38]:
```
t.count(3)
```

Out[38]:
```
3
```

In [39]:
```
t.count(4)
```

Out[39]:
```
2
```

In [40]:
```
t.count('word')# str
```

Out[40]:
```
1
```

```
In [41]: t.index("word")

Out[41]: 3
```

```
In [42]: t.index(9.3) # 8th location

Out[42]: 8
```

```
In [ ]: # immutable
```

List

- it is also heterogeneous data structure
- mutable in nature
- list() is the pre-defined function that represents the list
- [] square brackets
- list methods
  1. append
  2. count
  3. copy
  4. clear
  5. extend
  6. sort
  7. reverse
  8. pop
  9. remove
  10. insert

```
In [1]: # list initialization
        nums=input().split()
        print(nums)

        90 45 ruthu vanitha 90 34 56 nandini
        ['90', '45', 'ruthu', 'vanitha', '90', '34', '56', 'nandini']
```

In [27]:
```
# static list
li=[2,3,'python','workshop',90.34,3+2j,bin(int(input())),None,2,3,10,11,8,'apssdc
li
```

```
10
```

Out[27]:
```
[2,
 3,
 'python',
 'workshop',
 90.34,
 (3+2j),
 '0b1010',
 None,
 2,
 3,
 10,
 11,
 8,
 'apssdc']
```

In [28]:
```
li.index(2)
```

Out[28]:
```
0
```

In [29]:
```
li.remove(90.34)
li
```

Out[29]:
```
[2, 3, 'python', 'workshop', (3+2j), '0b1010', None, 2, 3, 10, 11, 8, 'apssdc']
```

In [30]:
```
li.remove(li[3])
li
```

Out[30]:
```
[2, 3, 'python', (3+2j), '0b1010', None, 2, 3, 10, 11, 8, 'apssdc']
```

In [31]:
```
li.extend([1,2,3]) # expands the list
li
```

Out[31]:
```
[2, 3, 'python', (3+2j), '0b1010', None, 2, 3, 10, 11, 8, 'apssdc', 1, 2, 3]
```

In [32]:
```
li.append([1,2,3]) # another data structure
li
```

Out[32]:
```
[2,
 3,
 'python',
 (3+2j),
 '0b1010',
 None,
 2,
 3,
 10,
 11,
 8,
 'apssdc',
 1,
 2,
 3,
 [1, 2, 3]]
```

In [33]:
```
li.insert(4,'siva') # index,value,if we execute twice or multiple time the same c
li
```

Out[33]:
```
[2,
 3,
 'python',
 (3+2j),
 'siva',
 '0b1010',
 None,
 2,
 3,
 10,
 11,
 8,
 'apssdc',
 1,
 2,
 3,
 [1, 2, 3]]
```

In [34]:
```
li.pop() # remove the last item by default
```

Out[34]:
```
[1, 2, 3]
```

In [35]:
```
li.pop(3)
```

Out[35]:
```
(3+2j)
```

In [ ]:
```
# disadv
# add,delete--->can't be updated
# list allows the duplicate items
```

In [36]:
```
li
```

Out[36]:
```
[2, 3, 'python', 'siva', '0b1010', None, 2, 3, 10, 11, 8, 'apssdc', 1, 2, 3]
```

In [39]:
```
# unique list of elements
unq=[]
for item in li:
    if item not in unq:
        unq.append(item)
print(unq)
```

```
[2, 3, 'python', 'siva', '0b1010', None, 10, 11, 8, 'apssdc', 1]
```

SET

- A well defined Collection of objects
- it is also heterogeneous data structure
- set()
- represented by{}
- mutable in nature

In [40]:
```
print(dir(set))
```

```
['__and__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '_
_dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt
__', '__hash__', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isu
b__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new_
_', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__',
'__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__s
ubclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_up
date', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'issubse
t', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_differenc
e_update', 'union', 'update']
```

In [41]:
```
A={8,4,9,10,23,54,1,9,5,10,45,90,12,9,14}
A
```

Out[41]:
```
{1, 4, 5, 8, 9, 10, 12, 14, 23, 45, 54, 90}
```

In [42]:
```
A.add(20)
A
```

Out[42]:
```
{1, 4, 5, 8, 9, 10, 12, 14, 20, 23, 45, 54, 90}
```

In [43]:
```
B={4,5,7,10,9,12,15,20}
B
```
Out[43]:
```
{4, 5, 7, 9, 10, 12, 15, 20}
```

In [44]:
```
A-B # deletes the values of B present in A
```
Out[44]:
```
{1, 8, 14, 23, 45, 54, 90}
```

In [45]:
```
A.difference(B)
```
Out[45]:
```
{1, 8, 14, 23, 45, 54, 90}
```

In [46]:
```
A.union(B)
```
Out[46]:
```
{1, 4, 5, 7, 8, 9, 10, 12, 14, 15, 20, 23, 45, 54, 90}
```

In [47]:
```
A.intersection(B)
```
Out[47]:
```
{4, 5, 9, 10, 12, 20}
```

In [48]:
```
A.isdisjoint(B)# returns true if sets doesn't have common elements
```
Out[48]:
```
False
```

In [49]:
```
A.symmetric_difference(B)# non-similar elements in both set
```
Out[49]:
```
{1, 7, 8, 14, 15, 23, 45, 54, 90}
```

In [50]:
```
A.issuperset(B)
```
Out[50]:
```
False
```

In [51]:
```
B.issubset(A)
```
Out[51]:
```
False
```

In [52]:
```
A.intersection_update(B)
A
```
Out[52]:
```
{4, 5, 9, 10, 12, 20}
```

In [53]:
```
B
```
Out[53]:
```
{4, 5, 7, 9, 10, 12, 15, 20}
```

In [54]:
```
A.symmetric_difference(B)
```
Out[54]:
```
{7, 15}
```

```
In [69]: A.difference_update(B)
         A

Out[69]: {1, 2, 6, 'apssdc', 'hi', 'new'}

In [70]: A.update(B)
         A

Out[70]: {1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}

In [71]: A.update({1,2,'new','apssdc'})
         A

Out[71]: {1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}

In [72]: A.update([4,5,6,'hi'])
         A

Out[72]: {1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}

In [73]: ord('1')

Out[73]: 49

In [74]: chr(10)

Out[74]: '\n'

In [75]: chr(2)

Out[75]: '\x02'

In [76]: # simply convert list into set if you want to remove duplicates
         print(li)

         [2, 3, 'python', 'siva', '0b1010', None, 2, 3, 10, 11, 8, 'apssdc', 1, 2, 3]

In [77]: new=[1,2,4,67,9,2,3,4,10]
         print(new)

         [1, 2, 4, 67, 9, 2, 3, 4, 10]

In [78]: set(new)

Out[78]: {1, 2, 3, 4, 9, 10, 67}

In [79]: A.discard(8) # non existed value
         A

Out[79]: {1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}
```

In [81]:
```
A.remove(8)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [81], in <cell line: 1>()
----> 1 A.remove(8)

KeyError: 8
```

Dictionary

- it is a paired data structure
- represented by{key:value}
- dict() is the pre-defined function
- dynamic data structure/mutable
    - keys can be any datatypes
        1. keys should be unique
        2. keys will act as index/reference
    - values can be any other data structure
        1. values might be similar
- key&value together called as item

In [82]:
```
print(dir(dict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem_
_', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior_
_', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '__setatt
r__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'cop
y', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'updat
e', 'values']
```

In [84]:
```
marks=[90,89,67,85]
dic={1:'hi','name':'student','friends':('ruthu','vanitha'),'subjects':marks,90.4
dic
# physical dict,search engine
```

Out[84]:
```
{1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point'}
```

In [ ]:
```
# working with dictionary
# method
```

In [85]:
```
print(dic.keys())# list of keys
```
```
dict_keys([1, 'name', 'friends', 'subjects', 90.45])
```

In [86]:
```
dic.values()
```
Out[86]:
```
dict_values(['hi', 'student', ('ruthu', 'vanitha'), [90, 89, 67, 85], 'point'])
```

In [87]:
```
dic.items()# list of tuple of items
```
Out[87]:
```
dict_items([(1, 'hi'), ('name', 'student'), ('friends', ('ruthu', 'vanitha')),
('subjects', [90, 89, 67, 85]), (90.45, 'point')])
```

In [ ]:
```
# entire dict depends only on keys
```

In [88]:
```
st='srkit'
for ch in st:
    print(ch)
```
```
s
r
k
i
t
```

In [89]:
```
for i in range(len(st)):
    print(st[i])
```
```
s
r
k
i
t
```

In [90]:
```
for each in dic:
    print(each)# you will get key values
```
```
1
name
friends
subjects
90.45
```

In [91]:
```
for key in dic:
    print(dic[key])# dic[key]=value
```
```
hi
student
('ruthu', 'vanitha')
[90, 89, 67, 85]
point
```

In [92]:
```
for item in dic.items():
    print(item)
```

```
(1, 'hi')
('name', 'student')
('friends', ('ruthu', 'vanitha'))
('subjects', [90, 89, 67, 85])
(90.45, 'point')
```

In [93]:
```
dic.fromkeys(marks)#creates a new dictionary with keys you pass
```

Out[93]:
```
{90: None, 89: None, 67: None, 85: None}
```

In [96]:
```
dic.setdefault('student')
dic
```

Out[96]:
```
{1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point',
 'student': None}
```

In [97]:
```
dic.update({'org':"apssdc"})
```

In [98]:
```
dic
```

Out[98]:
```
{1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point',
 'student': None,
 'org': 'apssdc'}
```

In [99]:
```
new={3:'hey',2:'hello'}
```

In [100]:
```
dic.update(new)
dic
```

Out[100]:
```
{1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point',
 'student': None,
 'org': 'apssdc',
 3: 'hey',
 2: 'hello'}
```

In [103]:
```
# prepare a dict of squares of numbers present in a range
sqs={}
for num in range(int(input()),int(input())):
    sqs[num]=num**2
print(sqs)
```

```
10
50
{10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225, 16: 256, 17: 289, 18: 32
4, 19: 361, 20: 400, 21: 441, 22: 484, 23: 529, 24: 576, 25: 625, 26: 676, 27:
729, 28: 784, 29: 841, 30: 900, 31: 961, 32: 1024, 33: 1089, 34: 1156, 35: 122
5, 36: 1296, 37: 1369, 38: 1444, 39: 1521, 40: 1600, 41: 1681, 42: 1764, 43: 18
49, 44: 1936, 45: 2025, 46: 2116, 47: 2209, 48: 2304, 49: 2401}
```

In [2]:
```
#prepare a dict of chars whose ascii values are even
chars={}
for ch in input():
    if ord(ch)%2==0:
        chars[ch]=ord(ch)
print(chars)
```

```
vijayawada
{'v': 118, 'j': 106, 'd': 100}
```

In [1]:
```
# string.format() method
name,loc=input(),input()
print("Myself {0} and I am from {1}".format(name,loc))
```

```
siva
vij
Myself siva and I am from vij
```

modules in python

- set of statements written to perform task said to be function
- group of functions callled as module
- group of modules called as packages

In [ ]:
```
# math module
```

In [4]:
```
import math
```

In [5]:
```
print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acos
h', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'co
s', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'fac
torial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isc
lose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'pro
d', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trun
c', 'ulp']
```

In [6]:
```
math.factorial(5)
```

Out[6]:
```
120
```

In [7]:
```
math.gcd(93,6)
```

Out[7]:
```
3
```

In [9]:
```
math.pow(6,7)
```

Out[9]:
```
279936.0
```

In [10]:
```
import random
```

In [11]:
```
print(dir(random))
```

```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'Syste
mRandom', 'TWOPI', '_Sequence', '_Set', '__all__', '__builtins__', '__cached_
_', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
'_accumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_floor', '_i
nst', '_log', '_os', '_pi', '_random', '_repeat', '_sha512', '_sin', '_sqrt',
'_test', '_test_generator', '_urandom', '_warn', 'betavariate', 'choice', 'choi
ces', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'logno
rmvariate', 'normalvariate', 'paretovariate', 'randbytes', 'randint', 'random',
'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform',
'vonmisesvariate', 'weibullvariate']
```

In [12]:
```
random.randint(1,60)
```

Out[12]:
```
42
```

In [1]:
```
import packages
```

In [2]:
```
print(dir(packages))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '_
_spec__']
```

In [3]:
```
from packages import functions,second
```

```
In [4]: dir(functions)

Out[4]: ['__builtins__',
         '__cached__',
         '__doc__',
         '__file__',
         '__loader__',
         '__name__',
         '__package__',
         '__spec__',
         'is_even',
         'is_perfect',
         'is_prime']
```

```
In [5]: dir(second)

Out[5]: ['__builtins__',
         '__cached__',
         '__doc__',
         '__file__',
         '__loader__',
         '__name__',
         '__package__',
         '__spec__',
         'frequency',
         'is_palindrome',
         'length']
```

```
In [6]: functions.is_even(8)

Out[6]: True
```

```
In [7]: functions.is_even(99)

Out[7]: False
```

```
In [8]: second.is_palindrome(input())

        121

Out[8]: True
```

```
In [33]: second.frequency('vijayawada','a')

Out[33]: 4
```

```
In [10]: functions.is_prime(13)

Out[10]: True
```

In [11]:
```
functions.is_perfect(15)
```

Out[11]:
```
False
```

Data Analysis

- now a day,data is big in size
- everyone is creating the data and using data
- the complete study of data is called data science
    - data analysis,machine learning,AI--DS
-

Data Science Modules

- numpy,pandas,searborn,matplotlib,open CV,scikit learn etc..

```
#### Numpy
- one of the data science modules
- Numpy stands for Numerical Python
- Used for scientific computations
- deals with array type of data
- homogeneous data structure
- cannot be modified
    - matrix ---array
- **module installation
    -
```

array()

- array() is the sub module of Numpy used to store homogeneous data items
- we can create upto 32 dimensional arrays
- numpy.array(data)

In [12]:
```
import numpy as n
```

In [13]:
```
print(dir(n))
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'Bytes0', 'CLIP', 'ComplexWarning',
'DataSource', 'Datetime64', 'ERR_CALL', 'ERR_DEFAULT', 'ERR_IGNORE', 'ERR_LOG',
'ERR_PRINT', 'ERR_RAISE', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZE
RO', 'FPE_INVALID', 'FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinit
y', 'MAXDIMS', 'MAY_SHARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecat
ionWarning', 'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'RAISE', 'RankWarn
ing', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLO
W', 'ScalarType', 'Str0', 'Tester', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_DEF
AULT', 'UFUNC_PYVALS_NAME', 'Uint64', 'VisibleDeprecationWarning', 'WRAP', '_No
Value', '_UFUNC_API', '__NUMPY_SETUP__', '__all__', '__builtins__', '__cached_
_', '__config__', '__deprecated_attrs__', '__dir__', '__doc__', '__expired_func
tions__', '__file__', '__getattr__', '__git_version__', '__loader__', '__mkl_ve
rsion__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_a
dd_newdoc_ufunc', '_distributor_init', '_financial_names', '_globals', '_mat',
'_pytesttester', '_version', 'abs', 'absolute', 'add', 'add_docstring', 'add_ne
wdoc', 'add_newdoc_ufunc', 'alen', 'all', 'allclose', 'alltrue', 'amax', 'ami
n', 'angle', 'any', 'append', 'apply_along_axis', 'apply_over_axes', 'arange',
'arccos', 'arccosh', 'arcsin', 'arcsinh', 'arctan', 'arctan2', 'arctanh', 'argm
ax', 'argmin', 'argpartition', 'argsort', 'argwhere', 'around', 'array', 'array
2string', 'array_equal', 'array_equiv', 'array_repr', 'array_split', 'array_st
r', 'asanyarray', 'asarray', 'asarray_chkfinite', 'ascontiguousarray', 'asfarra
y', 'asfortranarray', 'asmatrix', 'asscalar', 'atleast_1d', 'atleast_2d', 'atle
ast_3d', 'average', 'bartlett', 'base_repr', 'binary_repr', 'bincount', 'bitwis
e_and', 'bitwise_not', 'bitwise_or', 'bitwise_xor', 'blackman', 'block', 'bma
t', 'bool8', 'bool_', 'broadcast', 'broadcast_arrays', 'broadcast_shapes', 'bro
adcast_to', 'busday_count', 'busday_offset', 'busdaycalendar', 'byte', 'byte_bo
unds', 'bytes0', 'bytes_', 'c_', 'can_cast', 'cast', 'cbrt', 'cdouble', 'ceil',
'cfloat', 'char', 'character', 'chararray', 'choose', 'clip', 'clongdouble', 'c
longfloat', 'column_stack', 'common_type', 'compare_chararrays', 'compat', 'com
plex128', 'complex64', 'complex_', 'complexfloating', 'compress', 'concatenat
e', 'conj', 'conjugate', 'convolve', 'copy', 'copysign', 'copyto', 'core', 'cor
rcoef', 'correlate', 'cos', 'cosh', 'count_nonzero', 'cov', 'cross', 'csingle',
'ctypeslib', 'cumprod', 'cumproduct', 'cumsum', 'datetime64', 'datetime_as_stri
ng', 'datetime_data', 'deg2rad', 'degrees', 'delete', 'deprecate', 'deprecate_w
ith_doc', 'diag', 'diag_indices', 'diag_indices_from', 'diagflat', 'diagonal',
'diff', 'digitize', 'disp', 'divide', 'divmod', 'dot', 'double', 'dsplit', 'dst
ack', 'dtype', 'e', 'ediff1d', 'einsum', 'einsum_path', 'emath', 'empty', 'empt
y_like', 'equal', 'errstate', 'euler_gamma', 'exp', 'exp2', 'expand_dims', 'exp
m1', 'extract', 'eye', 'fabs', 'fastCopyAndTranspose', 'fft', 'fill_diagonal',
'find_common_type', 'finfo', 'fix', 'flatiter', 'flatnonzero', 'flexible', 'fli
p', 'fliplr', 'flipud', 'float16', 'float32', 'float64', 'float_', 'float_powe
r', 'floating', 'floor', 'floor_divide', 'fmax', 'fmin', 'fmod', 'format_float_
positional', 'format_float_scientific', 'format_parser', 'frexp', 'frombuffer',
'fromfile', 'fromfunction', 'fromiter', 'frompyfunc', 'fromregex', 'fromstrin
g', 'full', 'full_like', 'gcd', 'generic', 'genfromtxt', 'geomspace', 'get_arra
y_wrap', 'get_include', 'get_printoptions', 'getbufsize', 'geterr', 'geterrcal
l', 'geterrobj', 'gradient', 'greater', 'greater_equal', 'half', 'hamming', 'ha
nning', 'heaviside', 'histogram', 'histogram2d', 'histogram_bin_edges', 'histog
ramdd', 'hsplit', 'hstack', 'hypot', 'i0', 'identity', 'iinfo', 'imag', 'in1d',
'index_exp', 'indices', 'inexact', 'inf', 'info', 'infty', 'inner', 'insert',
'int0', 'int16', 'int32', 'int64', 'int8', 'int_', 'intc', 'integer', 'interp',
'intersect1d', 'intp', 'invert', 'is_busday', 'isclose', 'iscomplex', 'iscomple
xobj', 'isfinite', 'isfortran', 'isin', 'isinf', 'isnan', 'isnat', 'isneginf',
'isposinf', 'isreal', 'isrealobj', 'isscalar', 'issctype', 'issubclass_', 'issu
bdtype', 'issubsctype', 'iterable', 'ix_', 'kaiser', 'kron', 'lcm', 'ldexp', 'l
eft_shift', 'less', 'less_equal', 'lexsort', 'lib', 'linalg', 'linspace', 'litt
le_endian', 'load', 'loads', 'loadtxt', 'log', 'log10', 'log1p', 'log2', 'logad
dexp', 'logaddexp2', 'logical_and', 'logical_not', 'logical_or', 'logical_xor',
'logspace', 'longcomplex', 'longdouble', 'longfloat', 'longlong', 'lookfor', 'm
a', 'mafromtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matrix', 'matrixli
b', 'max', 'maximum', 'maximum_sctype', 'may_share_memory', 'mean', 'median',
'memmap', 'meshgrid', 'mgrid', 'min', 'min_scalar_type', 'minimum', 'mintypecod
e', 'mkl', 'mod', 'modf', 'moveaxis', 'msort', 'multiply', 'nan', 'nan_to_num',
'nanargmax', 'nanargmin', 'nancumprod', 'nancumsum', 'nanmax', 'nanmean', 'nanm
edian', 'nanmin', 'nanpercentile', 'nanprod', 'nanquantile', 'nanstd', 'nansu
m', 'nanvar', 'nbytes', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'ndinde
x', 'nditer', 'negative', 'nested_iters', 'newaxis', 'nextafter', 'nonzero', 'n
ot_equal', 'numarray', 'number', 'obj2sctype', 'object0', 'object_', 'ogrid',
'oldnumeric', 'ones', 'ones_like', 'os', 'outer', 'packbits', 'pad', 'partitio
n', 'percentile', 'pi', 'piecewise', 'place', 'poly', 'poly1d', 'polyadd', 'pol
yder', 'polydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub', 'po
lyval', 'positive', 'power', 'printoptions', 'prod', 'product', 'promote_type
s', 'ptp', 'put', 'put_along_axis', 'putmask', 'quantile', 'r_', 'rad2deg', 'ra
dians', 'random', 'ravel', 'ravel_multi_index', 'real', 'real_if_close', 'rec',
'recarray', 'recfromcsv', 'recfromtxt', 'reciprocal', 'record', 'remainder', 'r
epeat', 'require', 'reshape', 'resize', 'result_type', 'right_shift', 'rint',
'roll', 'rollaxis', 'roots', 'rot90', 'round', 'round_', 'row_stack', 's_', 'sa
fe_eval', 'save', 'savetxt', 'savez', 'savez_compressed', 'sctype2char', 'sctyp
eDict', 'sctypes', 'searchsorted', 'select', 'set_numeric_ops', 'set_printoptio
ns', 'set_string_function', 'setbufsize', 'setdiff1d', 'seterr', 'seterrcall',
'seterrobj', 'setxor1d', 'shape', 'shares_memory', 'short', 'show_config', 'sig
n', 'signbit', 'signedinteger', 'sin', 'sinc', 'single', 'singlecomplex', 'sin
h', 'size', 'sometrue', 'sort', 'sort_complex', 'source', 'spacing', 'split',
'sqrt', 'square', 'squeeze', 'stack', 'std', 'str0', 'str_', 'string_', 'subtra
ct', 'sum', 'swapaxes', 'sys', 'take', 'take_along_axis', 'tan', 'tanh', 'tenso
rdot', 'test', 'testing', 'tile', 'timedelta64', 'trace', 'tracemalloc_domain',
'transpose', 'trapz', 'tri', 'tril', 'tril_indices', 'tril_indices_from', 'trim
_zeros', 'triu', 'triu_indices', 'triu_indices_from', 'true_divide', 'trunc',
'typecodes', 'typename', 'ubyte', 'ufunc', 'uint', 'uint0', 'uint16', 'uint32',
'uint64', 'uint8', 'uintc', 'uintp', 'ulonglong', 'unicode_', 'union1d', 'uniqu
e', 'unpackbits', 'unravel_index', 'unsignedinteger', 'unwrap', 'use_hugepage',
'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version', 'void', 'void0', 'vs
plit', 'vstack', 'warnings', 'where', 'who', 'zeros', 'zeros_like']
```

In [ ]:
```
# creating array using str/tuple/list/dict/set
```

In [14]:
```
st='vijayawada'# string-object
n.array(st)
```

Out[14]:
```
array('vijayawada', dtype='<U10')
```

In [17]:
```
# convert tuple into array
tp=(4,5,6,'hi','hello')
ar=n.array(tp)
print(ar)
```

```
['4' '5' '6' 'hi' 'hello']
```

In [18]:
```
li=[3,4,6,7,89,90]
ar=n.array(li)
print(ar)
```

```
[ 3  4  6  7 89 90]
```

In [19]:
```
# conversion of set into array
print(n.array({2,3,4,5,6,8,9,1,2,4,0,10,}))
```

```
{0, 1, 2, 3, 4, 5, 6, 8, 9, 10}
```

In [21]:
```
# conversion of dictionary into array
dic
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [21], in <cell line: 2>()
      1 # conversion of dictionary into array
----> 2 dic

NameError: name 'dic' is not defined
```

In [22]:
```
dic={1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point',
 'student': None,
 'org': 'apssdc'}
```

In [23]:
```
dic
```

Out[23]:
```
{1: 'hi',
 'name': 'student',
 'friends': ('ruthu', 'vanitha'),
 'subjects': [90, 89, 67, 85],
 90.45: 'point',
 'student': None,
 'org': 'apssdc'}
```

In [25]:
```
ar=n.array(dic)
ar
```

Out[25]:
```
array({1: 'hi', 'name': 'student', 'friends': ('ruthu', 'vanitha'), 'subjects':
[90, 89, 67, 85], 90.45: 'point', 'student': None, 'org': 'apssdc'},
      dtype=object)
```

In [26]:
```
# range
n.array(range(15))
```

Out[26]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [27]:
```
print(n.array(range(10,50)))
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
```

In [29]:
```
print(n.array(range(1,50,6),dtype='float'))
```

```
[ 1.  7. 13. 19. 25. 31. 37. 43. 49.]
```

In [30]:
```
# some attributes
```

In [39]:
```
#list of tuple/list
ar1=n.array([[1,2,3],[3,4,7]],ndmin=3)
print(ar1)
```

```
[[[1 2 3]
  [3 4 7]]]
```

In [33]:
```
# size,itemsize,shape,ndim,ndmin
```

In [40]:
```
ar1
```

Out[40]:
```
array([[[1, 2, 3],
        [3, 4, 7]]])
```

In [41]:
```
ar1.size# no.of elements
```

Out[41]:
```
6
```

In [42]:
```
ar1.shape
```

Out[42]:
```
(1, 2, 3)
```

In [43]:
```
ar1.itemsize# data size
```

Out[43]:
```
4
```

In [44]:
```
ar1.ndim # no.of dimensions
```

Out[44]:
```
3
```

In [45]:
```
mul=n.array([1,2,3,8,4,5,10],ndmin=5)
mul
```

Out[45]:
```
array([[[[[ 1,  2,  3,  8,  4,  5, 10]]]]])
```

In [46]:
```
mul.ndim
```
Out[46]:
```
5
```

In [47]:
```
# zero matrix
# one
#full
#fill
#diag
#eye
```

In [48]:
```
z=n.zeros(4)
z
```
Out[48]:
```
array([0., 0., 0., 0.])
```

In [49]:
```
z=n.zeros((4,3))
z
```
Out[49]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [50]:
```
o=n.ones((3,4),dtype=int)
o
```
Out[50]:
```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])
```

In [51]:
```
# identity
i=n.eye(4)
i
```
Out[51]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [52]:
```
i=n.eye(4,5)
i
```
Out[52]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

In [55]:
```
# full and fill
f1=n.full((4,3),5)
f1
```

Out[55]:
```
array([[5, 5, 5],
       [5, 5, 5],
       [5, 5, 5],
       [5, 5, 5]])
```

In [56]:
```
f1.fill(2)
f1
```

Out[56]:
```
array([[2, 2, 2],
       [2, 2, 2],
       [2, 2, 2],
       [2, 2, 2]])
```

In [57]:
```
dg=n.diag([3,4,5,9])
dg
```

Out[57]:
```
array([[3, 0, 0, 0],
       [0, 4, 0, 0],
       [0, 0, 5, 0],
       [0, 0, 0, 9]])
```

In [ ]: