Today Topics

- Data Structures
- DA Modules

Data Structure

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

tuple

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

In [139]:
```
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 [140]:
```
for item in tp:
    print(item)
```

```
1
2
word
90.45
python
(4+5j)
True
0b110
```

In [141]:
```
#using index
tp[3] #4th element present in 3rd index
```

Out[141]: 90.45

In [142]:
```
tp[-1] # last ele
```

Out[142]: '0b110'

In [143]:
```
tp[-2] # last but one
```

Out[143]: True

In [144]:
```
# slicing means extracting some part of iterabl
# using index
```

In [145]:
```
tp[:] #
```

Out[145]: (1, 2, 'word', 90.45, 'python', (4+5j), True, '0b110')

In [146]:
```
tp[::]
```

Out[146]: (1, 2, 'word', 90.45, 'python', (4+5j), True, '0b110')

In [147]:
```
tp[::-1] # reversed itrerable
```

Out[147]: ('0b110', True, (4+5j), 'python', 90.45, 'word', 2, 1)

In [148]:
```
tp[::2] # alternate values
```

Out[148]: (1, 'word', 'python', True)

In [149]:
```
tp[::-2] # alternate items in the reverse order
```

Out[149]: ('0b110', (4+5j), 90.45, 2)

In [150]:
```
tp[2:5] # upper bound is exclusive
```

Out[150]: ('word', 90.45, 'python')

In [151]:
```
tp[:4] # starts from first by default
```

Out[151]: (1, 2, 'word', 90.45)

In [152]:
```
tp[3:] # up to the end
```

Out[152]: (90.45, 'python', (4+5j), True, '0b110')

In [153]:
```
bin(9)# binary format of value also stored in str
```

Out[153]:
```
'0b1001'
```

In [154]:
```
count() # frequency of item
# no.of occurence of data item
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [154], in <cell line: 1>()
----> 1 count()

TypeError: 'int' object is not callable
```

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

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

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

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

In [ ]:
```
# find the frequency of 3
count=0
for val in t:
    if val==4:
        count+=1
print(count)

```

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

In [ ]:
```
t.count(4) # number
```

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

In [ ]:
```
t.index('word')
```

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

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

list

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

In [ ]:
```
dir(list)
```

In [ ]:
```
# list intirialization
nums=input().split() # dynamic str list
print(nums)
```

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

In [ ]:
```
print(li)
```

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

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

In [ ]:
```
li
```

In [ ]:
```
li.remove(90.34) # li[3]
```

In [ ]:
```
li
```

In [ ]:
```
li[type(complex)]  # index
```

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

In [ ]:
```
li
```

In [ ]:
```
li.extend([1,2,3])
```

In [ ]:
```
print(li) # expands the list
```

In [ ]:
```
li.insert(4,'new') # index,value
```

In [ ]:
```
li
```

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

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

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

```

In [ ]:
```
li
```

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

SET

- A well defined collection of objects
- it is also heterogenous data structure

- set()
- represented by {}
- mutable in nature

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

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

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

In [ ]:
```
A
```

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

In [ ]:
```
A-B # deletes the values of B present in A
```

In [ ]:
```
A.difference(B)
```

In [ ]:
```
A.union( B)
```

In [ ]:
```
A.intersection(B)
```

In [ ]:
```
A.isdisjoint(B) # returns True if sets doesn't
# have common ele
```

In [ ]:
```
A.symmetric_difference(B)
#non-similar elements in both set
```

In [ ]:
```
A.issuperset(B)
```

In [155]:
```
B.issubset(A)
```

Out[155]:
```
False
```

In [156]:
```
A.intersection_update(B)
```

In [157]:
```
A
```

Out[157]:
```
{4, 5, 7, 10, 12, 15}
```

In [158]:
```
1 B
```

Out[158]: {4, 5, 7, 9, 10, 12, 15, 20}

In [159]:
```
1 dir(set)
```

Out[159]:
```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
```

```
'union',
'update']
```

In [160]:
```
A.difference_update(B)
```

In [161]:
```
A # empty
```

Out[161]:
```
set()
```

In [162]:
```
A.update(B)
```

In [163]:
```
A
```

Out[163]:
```
{4, 5, 7, 9, 10, 12, 15, 20}
```

In [164]:
```
A.update({1,2,'new','apssdc'})
```

In [165]:
```
A
```

Out[165]:
```
{1, 10, 12, 15, 2, 20, 4, 5, 7, 9, 'apssdc', 'new'}
```

In [166]:
```
A.update([4,5,6,'hi'])
```

In [167]:
```
A
```

Out[167]:
```
{1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}
```

In [168]:
```
ord('1')
```

Out[168]:
```
49
```

In [169]:
```
chr(10)
```

Out[169]:
```
'\n'
```

In [170]:
```
chr(2)
```

Out[170]:
```
'\x02'
```

In [171]:
```
# simpy convert list into set if you want to remove
# duplicates
```

In [172]:
```
print(li)
```

```
[2, 3, 'python', 'new', (3+2j), '0b1000', None, 2, 3, 10, 11, 8, 'apssdc', [1,
2, 3], 1, 2, 3]
```

In [173]:
```
new=[1,2,4,67,9,2,3,4,10]
print(new)
```

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

In [174]:
```
set(new)
```

Out[174]:
```
{1, 2, 3, 4, 9, 10, 67}
```

In [175]:
```
A
```

Out[175]:
```
{1, 10, 12, 15, 2, 20, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}
```

In [176]:
```
A.remove(20)
```

In [177]:
```
A
```

Out[177]:
```
{1, 10, 12, 15, 2, 4, 5, 6, 7, 9, 'apssdc', 'hi', 'new'}
```

In [178]:
```
A.discard(9)
```

In [179]:
```
A
```

Out[179]:
```
{1, 10, 12, 15, 2, 4, 5, 6, 7, 'apssdc', 'hi', 'new'}
```

In [180]:
```
A.discard(8) # non existed
```

In [181]:
```
A
```

Out[181]:
```
{1, 10, 12, 15, 2, 4, 5, 6, 7, 'apssdc', 'hi', 'new'}
```

In [182]:
```
A.remove(8)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [182], in <cell line: 1>()
----> 1 A.remove(8)

KeyError: 8
```

Dictionary

- it is a paired data structure
- represented by {key:value}
- dict() is the predefined function
- dynamic data structure/mutable

- keys can be any datatype
  1. keys should be unique
  2. key will act as index/reference
- values can be any other data structure
  1. values might be similar
- key&value together called as item

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

In [183]:
```
marks=[90,89,67,85]
dic={1:'hi','name':'student',
    'friends':('ruthu','vanitha'),
    'subjects':marks,90.45:'point'}
print(dic)
# physical dict,search
```

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

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

In [184]:
```
dic.values() # list of values
```

Out[184]:
```
dict_values(['hi', 'student', ('ruthu', 'vanitha'), [90, 89, 67, 85], 'point'])
```

In [ ]:
```
print(dic.keys()) # list of keys
```

In [ ]:
```
dic.items() # list of tuple of items
```

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

In [ ]:
```
st='srkit'
for ch in st:
    print(ch)
```

In [ ]:
```
for i in range(len(st)):
    print(st[i])
```

In [ ]:
```
for each in dic:
    print(each) # you will get key values
```

In [ ]:
```
for key in dic:
    print(dic[key]) # dic[key]=value
```

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

In [ ]:
```
help(dic.fromkeys)
```

In [185]:
```
dic.fromkeys(marks) # creates a new dictionary
# with keys you pass
```

Out[185]:
```
{90: None, 89: None, 67: None, 85: None}
```

In [ ]:
```
marks
```

In [ ]:
```

```

In [ ]:
```
help(dic.setdefault)
```

In [ ]:
```
dic.setdefault('student')
# None is allocated by default
```

In [ ]:
```
dic
```

In [ ]:
```

```

In [186]:
```
dic.update({'org':"apssdc"})
```

In [187]:
```
dic
```

...

In [188]:
```
new={3:'hey',2:'hello'}
new
```

...

In [189]:
```
dic.update(new)
```

In [190]:
```
dic
```

...

In [191]:
```
# prepare a dict of squares of numbers present
# in a range
```

In [192]:
```
# 5 to 15:{sqaures}
```

In [193]:
```
sqs={}
for num in range(int(input()),int(input())):
    sqs[num]=num**2
print(sqs)
```

...

In [194]:
```
# prepare a dict of chars whose ascii values are even
# vijayawada:ascii value and char=={}
chars={}
for ch in input():
    if ord(ch)%2==0:
        chars[ch]=ord(ch)
print(chars)
```

...

In [195]:
```
# string.format() method
name,loc=input(),input()
print("Myself {0} and I am from {1}".format(name,loc))
```

...

In [196]:
```
 # fstring/string.format
    # string.format() method
name,loc=input(),input()
print("Myself {} and I am from {}".format(name,loc))
```

...

modules

In [197]:
```
dic
```

...

In [198]:
```
dic['name']
```

...

In [199]:
```
dic['name']='siva'
```

In [200]:
```
dic
```

...

Modules in python

- set of statements written to perform task said to be function

- group of functions called as module
- group of modules called as package
  - engg-->branches(modules)-->years(functions)

working with modules

- installation of module in our local machine
  - pip install module
- using the module at our work space

In [201]:
```
# math module
```

In [204]:
```
import math
 # for computation
```

In [203]:
```
dir(math)
```

...

In [205]:
```
math.factorial(5)
```

...

In [208]:
```
math.gcd(93,71)
```

...

In [210]:
```
math.pi
```

...

In [212]:
```
math.pow(8,3)
```

...

In [213]:
```
import random
```

In [214]:
```
dir(random)
```

...

In [220]:
```
random.randint(1,60)
```

...

In [221]:
```
import package
```

In [222]:
```
dir(package)
```

...

In [224]:
```
from package import functions
```

In [225]:
```
from package import functions,second
```

In [226]:
```
dir(functions)
```

Out[226]:
```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'is_even',
 'is_perfect',
 'is_prime']
```

In [227]:
```
dir(second)
```

...

In [228]:
```
functions.is_even(8)
```

...

In [229]:
```
functions.is_even(11)
```

...

In [230]:
```
functions.is_prime(13)
```

...

In [231]:
```
functions.is_prime(25)
```

...

In [232]:
```
functions.is_perfect(6)
```

...

In [233]:
```
#usage of functions
for num in range(1,1000):
    if functions.is_perfect(num):
        print(num)
```

...

In [234]:
```
#28=1+2+4+7+14
```

In [235]:
```
second.is_palindrome(input())
```

...

In [236]:
```
second.is_palindrome("madam")
```

...

In [237]:
```
second.frequency('vijayawada','a')
```

...

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
- homogenous data structure
- cannot be modified
  - matrix --array
- **module installation**
  - pip install numpy
  - import Numpy as np

array()

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

In [239]:
```
import numpy as n
```

In [240]:
```
dir(n)
```

...

In [241]:
```
help(n.array)
```

...

In [242]:
```
# creating array using str/tuple/list/dict/set
```

In [243]:
```
st='vijayawada' # string-object
n.array(st)
```

...

In [246]:
```
# convert tuple into array
tp=(4,5,6,'hi','hello')
ar=n.array(tp)
print(ar)
```

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

In [247]:
```
li=[3,4,6,7,89,90]
ar=n.array(li)
print(ar)
```

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

In [248]:
```
# 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 [250]:
```
# conversion of dictionary into array
```

In [251]:
```
dic
```

...

In [253]:
```
ar=n.array(dic)
ar
```

...

In [255]:
```
# range
n.array(range(15))
```

...

In [257]:
```
print(n.array(range(10,50)))
```

...

In [260]:
```
print(n.array(range(1,50,6),dtype='float'))
```

...

In [261]:
```
# some attributes
```

In [262]:
```
# 2D arrays
```

In [281]:
```
# list of tuples\lists
arl=n.array([[1,2,4],[3,4,7]])
print(arl)
```

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

In [267]:
```
# size,itemsize,shape,ndim,ndmin
```

In [268]:
```
arl
```

...

In [269]:
```
arl.size # no.of elements
```

...

In [270]:
```
arl.shape
```

Out[270]: (2, 3)

In [271]:
```
arl.itemsize # data size
```

...

In [273]:
```
arl.ndim # no.of dimensions
```

...

In [278]:
```
arl.ndim
```

Out[278]: 2

In [282]:
```
mul=n.array([1,2,3,8,4,5,10],ndmin=5)
mul
```

...

In [283]:
```
# zeros matrix
# ones
# full
# fill
# diag
# eye

```

In [284]:
```
z=n.zeros(4)
z
```

...

In [286]:
```
z=n.zeros((4,3))
z
```

...

In [290]:
```
o=n.ones((3,4),dtype=int)
o
```

Out[290]:
```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])
```

In [291]:
```
# identity
i=n.eye(4)
i
```

...

In [292]:
```
# identity
i=n.eye(4,5)
i
```

...

In [303]:
```
# full and fill
fl=n.full((4,3),5)
fl
```

Out[303]:
```
array([[5, 5, 5],
       [5, 5, 5],
       [5, 5, 5],
       [5, 5, 5]])
```

In [298]:
```
help(n.full)
```

...

In [304]:
```
1 fl.fill(2)
2 fl
```

. . .

In [305]:
```
1 dg=n.diag([3,4,5,9])
2 dg
```

. . .

In [ ]:
```
1
```