Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Kmit
SONET

Session - 12

DATA SCIENCE

This session deals with

Why Data Structure

What is Data Structure

Where Data Structures are used

Types of Data Structures

Why Data Structures...?

As s data scientist, our job is to perform operations on data.

we basically perform the following three steps

1) Take some input

2) Process it

3) Give back the output.

The input can be in any form, Ex: while searching for directions on google maps, starting point and the destination as input to google maps

while logging in to facebook, email and password as input

The computer application gives us output.

Data structures refers to the way of organize the information on computer.

Why Data Structures...?

Suppose to have a book on Set Theory from a public library,
to do that you have to first go to the maths section, then to set theory section.

If these books are not organized in this manner and just distributed randomly then it will be really a cumbersome process to find a book on set theory.

Librarian organizes his books(data) into a particular form (data structure) to efficiently perform a task(find a book on set theory).

What is Data Structures...?

At the backbone of every program or piece of software are two entities: data and algorithms.

Algorithms transform data into something a program can effectively use.

Data structures are the way we are able to store and retrieve data.

Kmit
SONET

About Data Structure...!

DATA SCIENCE

Inputting

Maintaining

Data Structure can Handle

Processing

Retrieving

SONET

Types of Data Structures

DATA SCIENCE

Lists Data Structures

- **List.** Is a collection of items of same or different data types.
- A list stores elements one after another with comma separator.
- List is index based with index starting with zero.
- Elements are enclosed in square brackets [ ]
- Elements in the list can repeat
- It does not provide fast lookups.
- Finding an element is often slow.
- A search is required.

- List is collection of elements of different types:
- >>> list_mixed = [1,1.5,'A','KMIT Hyderabad']
- >>> print(list_mixed)
- [1, 1.5, 'A', “KMIT Hyderabad”]

- List can be collection of elements of same type
- >>> list = ['Ram', 'Shyam', 'Ravi', 'Kishor']
- >>> print(list)
- ['Ram', 'Shyam', 'Ravi', 'Kishor']
- String can be in single or double quotes
- >>> li = ["one", ‘two’,"three"]
- >>> print(li)
- ['one', 'two', 'three']

Exercise

- Write a program to create a list “items” and initialize it with 5 elements.
- Let first three elements be strings having the first element of the list repeating in the list and other two elements as integer and float values.
- Print the data type of each element in the list
- print(type(items[1]),type(items[4]),type(items[5]))

Exercise


```
items = [ "book", "computer", ‘book’,1,1.5]
for element in items:
        print(type(element))
```

Output

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'int'>
<class 'float'>
```

- It can have any number of items and they may be of different types(integers,float,string etc..)
- Syntax:
- My_list=[]
- Ex:
- #lists
- my_list=[1,2,3,4,5]
- print(my_list)
- #list with mixed type
- my_list1=[2,"python",3.65]
- print(my_list1)

- #nested lists
- my_list2=["hello",[2,3,4,5],[3,2.3,4.21]]
- print(my_list2)

- **Accessing List:**
- We can use the index operator[] to access an item in a list.
- **Ex:**
- **my_list2=["hello",[2,3,4,5],[3,2.3,4.21]]**
- **print(my_list2)**
- **for i in my_list2:**
- **print(i)**

Accessing list

```
>>a=[0,11,22,33,44,55,66,77,88,99]
Syntax of slicing: a[x:y:z]
>>> a[0:7:1]    #elements from 0 (included) to 7 (excluded)
[0, 11, 22, 33, 44, 55, 66]
>>> a[0:7:2]
[0, 22, 44, 66]
#If z is not given, z is taken as 1
>>> a[0:7]
[0, 11, 22, 33, 44, 55, 66]
#If z is positive , slicing will be from left to right , so x should be less than y.
>>> a[2:7:1]
[22, 33, 44, 55, 66]
>>> a[7:2:1]  #wrong indexes for slicing
[]
#If z is negative ,slicing will be from right to left, so x should be greater than y
>>> a[7:2:-1]
[77, 66, 55, 44, 33]
```

Accessing list

'''
if z is positive and if x is not given , x is taken as 0 if y is not given , y is taken as the length of the list
'''

```
>>> a[:7:1]
[0, 11, 22, 33, 44, 55, 66]
>>> a[5::1]
[55, 66, 77, 88, 99]
```

'''
if z is negative and if x is not given , x is taken as -1 (index of the last element)
        if y is not given , y is taken as negative of length of the list
'''

```
>>> a[:-5:-1]
[99, 88, 77, 66]
>>> a[-5::-1]
[55, 44, 33, 22, 11, 0]
>>> a[5:-1:1]
[55, 66, 77, 88]
>>> a[-1:5:-1]
[99, 88, 77, 66]
```

SONET

List-Functions DATA SCIENCE

- basic operations or functions
- Append()-add an elements to the end of the list
- Extend()-add all elements of a list to the another list
- Insert()-insert an item at the defined index
- Len()- returns number of elements
- Remove()-removes an item and returns an element at the given index
- Clear()-removes all item from the list
- Index()-returns the index of the first matched item
- Count()-returns the count of number of items passed as an argument
- Sort()-sort items in a list in ascending order
- Reverse()-reverse the order of items in the list
- Copy()-returns a shallow copy of the list
- Pop()-it wll remove the top element from the list

SONET

List-Functions DATA SCIENCE

- **Append:**This method is called upon the list instance (which must not equal None). It receives the value we are adding.
- **Append**
- list = []
- list.**append**(1)
- list.append(2)
- list.append(6)
- list.append(3)
- print(list)
- **Output**
- [1, 2, 6, 3]

Example

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3
mylist.append("Ram")
mylist.append('A')
mylist.append(15.56)
print(mylist[3]) # prints Ram
print(mylist[4]) # prints A
print(mylist[5]) # prints 15.56
```

Output

```
1
2
3
Ram
A
15.56
```

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3
mylist.append("Ram")
mylist.append('A')
mylist.append(15.56)
print(mylist[3]) # prints Ram
print(mylist[4]) # prints A
print(mylist[5]) # prints 15.56
Print(mylist[6])
```

Output

```
1
2
3
Ram
A
15.56
Traceback (most recent call last):
  File
"D:\UBG\Python\PythonCode\ListCode\
ListDemo1.py", line 14, in <module>
    print(mylist[6]) # prints 15.56
IndexError: list index out of range
```

- **Extend.**
- A list can be appended to another list with extend().
- We extend one list to include another list at its end.
- We concatenate (combine) lists.
- **Caution:**If we try to call append() to add a list, the entire new list will be added as a single element of the result list.
- **Tip:**Another option is to use a for-loop, or use the range syntax to concatenate (extend) the list.

Python List

- # Two lists.
- a = [1, 2, 3]
- b = [4, 5, 6]
- # Add all elements in list b to list a.
- **a.extend**(b)
- List a now contains six elements.
- print(a)
- **Output** [1, 2, 3, 4, 5, 6]

- **Insert.** An element can be added anywhere in a list. With insert() we can add to the first part or somewhere in the middle of the list.
- Lists are indexed starting at zero—they are zero-based.
- The index 1 indicates the second element location.
- Example:
- list = ["dot", "perls"]
- # Insert at index 1.
- list.**insert**(1, "net")
- print(list)

- **Len.** A list contains a certain number of elements. This may be zero if it is empty. With len, a built-in method, we access the element count.
- **Python program that uses len**
- animals = []
- animals.append("cat")
- animals.append("dog")
- count = **len**(animals)
- # Display the list and the
- length. print(animals)
- print(count)
- **Output**
- ['cat', 'dog']
- 2

Lists

- in keyword. Is an element in a list?
- We use "in" and "not in“ keywords as membership operators to determine:
- Is an element in a list?
- Other approaches are possible but "in" is simplest.
- Here we search a list with "in" and "not in."

```
items = [‘computer’,”atlas”,1,1.5,”,marks”]
if "computer" in items:
    print(1)
if "atlas" in items:
    # This is not reached.
    print(2)
else:
    print(3)
if "marker" not in items:
    print(4)
```

- **Sort, reverse.** Lists maintain the order of their elements. And they can be reordered.
- With the sort method, we change the order of elements from low to high.
- With reverse, we invert the current order of the elements.
- Sort and reverse can be combined (for a reversed sort).

Lists

- list = [400, 500, 100, 2000]
- # Reversed.
- list.**reverse**()
- print(list)
- # Sorted.
- list.**sort**()
- print(list)
- # Sorted and reversed.
- list.reverse()
- print(list)
- **Output**
- [2000, 100, 500, 400]
- [100, 400, 500, 2000]
- [2000, 500, 400, 100]

- **Remove, del.** Remove acts upon a value.
- It first searches for that value, and then removes it.
- Elements (at an index) can also be removed with the del statement.
- **Remove:**This takes away the first matching element in the list. We can call it multiple times, and it could keep changing the list.
- **Del:**This meanwhile removes elements by index, or a range of indices. Del uses the slice syntax.
- names = ["Tommy", "Bill", "Janet", "Bill", "Stacy"]

Lists

```
# Remove this value.
names.remove("Bill")
print(names)
# Delete all except first two elements.
del names[2:]
print(names)
# Delete all except last element.
del names[:1]
print(names)
```

Output ['Tommy', 'Janet', 'Bill', 'Stacy']
['Tommy', 'Janet']
['Janet']

Lists

- **Count.** This method does not return the number of elements in the list. Instead it counts a specific element. As an argument, pass the value of the element you wish to count.
- **Note:**Internally count() loops through all the elements and keeps track of the count. It then returns this value.

Lists

- **Python that uses count**
- names = ['a', 'a', 'b', 'c', 'a']
- # Count the letter a.
- value = names.**count**('a')
- print(value)
- **Output**
- 3

Lists

- **Index.** This searches lists.
- We pass it an argument that matches a value in the list.
- It returns the index where that value is found.
- If no value is found, it throws an error.
- **Tip:**For programs where you need more control, please consider using a for-loop instead of index().
- **Info:**With a for-loop, we can more elegantly handle cases where the value is not found. This avoids the ValueError.

Lists

- # Input list.
- values = ["uno", "dos", "tres", "cuatro"]
- # Locate string.
- n = values.**index**("dos")
- print(n, values[n])
- # Locate another string.
- n = values.**index**("tres")
- print(n, values[n])
- n = values.**index**("?")
- **Output**
- 1 dos
- 2 tres
- Not found

- **For-loop.** In list loops, we often need no index. We require just the elements in order. The for-loop is ideal in this situation. It eliminates the confusion of an index variable.
- **Here:**We encounter each of the four list elements in the for-loop. We use the identifier "element" in the loop body.

Lists

- # An input list.
- elements = ["spider", "moth", "butterfly", "lizard"]
- # Use simple for-loop.
- **for** ele in elements:
- print(ele)
- **Output**
- spider moth butterfly lizard

Lists

- **Min, max.** We do not need to search for the smallest or largest element in a list. Instead we use max and min. Internally this searches.
- **Here:**Max returns the value 1000, which is larger than all other elements in the list. And min returns negative -100.

List Data Structures

- values = [-100, 1, 10, 1000]
- # Find the max and min elements. print(**max**(values))
- print(**min**(values))

Exercise-1

- Create a list having names of any number of students and another list having marks of any number of subjects.
- Prompt the user to enter name of a student and then his marks.
- Display average marks of above student.
- Repeat above for all students.
- Finally display the highest average marks and the name of student

```
marks=[]
names=[]
avg1=[]
num=int(input("Enter the number of students:"))
num1=int(input("Enter the number of subjects:"))
for i in range(1,num+1):
    name=str(input("Name of student %d: " %i))
    names.append(name)
    marks=[]
    for j in range(1,num1+1):
        mark=int(input("Enter subject %d mark : " %j))
        marks.append(mark)
    tot=sum(int(j) for j in marks)
    print("Student " , i," marks are: ")
    print(marks)
```

Lists

```
print("Total marks:  " ,tot)
    avg=tot/num1
    avg1.append(avg)
    print("Average marks obtained:  ",avg)
    for k in range(len(marks)):
        if(marks[k]<40):
            print("Failed")
        else:
            print("Passed")
print("Highest average is: %d\n Name: %s\n"%(max(avg1),name))
```

List Comphrensions

- 1.find cube of each element in a list
- #practice
- l1=[2,3,1]
- cubes=[n*n*n for n in l1]
- print(l1)
- print(cubes)

- 2.find square root of each element in the list
- #practice
- import math
- l1=[4,9,1,49]
- sqt=[math.sqrt(n) for n in l1]
- print(l1)
- print(sqt)

Exercise-3

Find common numbers from two list using list comprehension

```
list_a = [1, 2, 3, 4]
list_b = [2, 3, 4, 5]
```

Solution

```
common_num = [a for a in list_a for b in list_b if a == b]

print(common_num)
```

Output

```
[2, 3, 4]
```

Exercise-4

Return numbers from the list which are not equal as tuple

```
l1 = [1, 2, 3]
l2 = [2, 7]


different_num = [(x, y) for x in l1 for y in l2 if a != b]

print(different_num) # Output: [(1, 2), (1, 7), (2, 7), (3, 2), (3, 7)]


#Output: [(1, 2), (1, 7), (2, 7), (3, 2), (3, 7)]
```

Exercise-5

Convert list elements to lower case

Solution

```
list_a = ["Hello", "World", "In", "Python"]

small_list_a = [str.lower() for str in list_a]

print(small_list_a)


 # Output: ['hello', 'world', 'in', 'python']
```

Exercise-6

1. Write a python program to create a list of even numbers and another list of odd numbers from a given list
2. Write a python program to remove repeated elements from a given list without using built-in methods
3. Write a python program to find the longest word in a given sentence
4. Write a python program to find number of occurrences of given number with out using built-in methods
5. ["www.zframez.com", "www.wikipedia.org", "www.asp.net", "www.abcd.in"]
Write a python program to print website suffixes (com , org , net ,in) from this list
6. Write a python program to sort a given list of numbers with out using sort() function

Conclusion

You are aware of

Data Structures

Types of Data Structures

We will proceed with

More on Data Structures

Kmit

SONET

DATA SCIENCE

THANK YOU