PythonCourse_5_Arrays&Lists

April 18, 2021

0.1 Arrays & Lists

[1]:
```
li = []
print(type(li))
```

```
<class 'list'>
```

[4]:
```
li = [1,2,'abc',6.5] #heterogeneous elements
```

[3]:
```
li
```

[3]:
```
[1, 2, 'abc', 6.5]
```

[6]:
```
len(li) #length of the list
```

[6]:
```
4
```

[7]:
```
li[0] #access through index
```

[7]:
```
1
```

[8]:
```
li[1] = 10 # change element at particular index
print(li)
```

```
[1, 10, 'abc', 6.5]
```

[17]:
```
## slice a list

print(li[:]) # prints all elements
print(li[1:])# prints id 1 to end
print(li[:3])# prints till before id 3
print(li[1:3])# prints from id 1 till before 3
print(li[1:10])# prints from id 1 till before 10--> prints till available when␣
 ↪that id is not there
print(li[:10])
```

```
[1, 10, 'abc', 6.5]
[10, 'abc', 6.5]
[1, 10, 'abc']
[10, 'abc']
```

```
[10, 'abc', 6.5]
[1, 10, 'abc', 6.5]
```

[9]:
```
## append to a list

li = [1,2,'abc',6.5]
li.append(1) # appends 1 at end
print(li)
#li.append() #needs  one arg
#li.append(1,3) #takes only one argument
li
```

```
[1, 2, 'abc', 6.5, 1]
```

[9]:
```
[1, 2, 'abc', 6.5, 1]
```

[13]:
```
## insert to a list

#li.insert('jay') #needs two args
li.insert(1, 'jay') #inserts jay at index
print(li)
li.insert(10,'rekhu') #inserts at end if index not present
print(li)
li.insert(2,['srini',9]) #inserts list element inside list and does not add␣
 ↪elements individually to the outer list
print(li)
```

```
[1, 'jay', 'jay', 'jay', 2, 'abc', 6.5, 1, 'rekhu']
[1, 'jay', 'jay', 'jay', 2, 'abc', 6.5, 1, 'rekhu', 'rekhu']
[1, 'jay', ['srini', 9], 'jay', 'jay', 2, 'abc', 6.5, 1, 'rekhu', 'rekhu']
```

[17]:
```
## extend a list

li = [1,2,'abc',6.5]
li.extend([1,2,3]) #adds these elements individully to the list at end
print(li)
#li.extend(2,[4,5]) #takes only one arg
#print(li)
```

```
[1, 2, 'abc', 6.5, 1, 2, 3]
```

[21]:
```
## remove element from list

li = [1, 2, 'abc', 6.5, 1, 2, 3]
li.remove(2) #removes 2; if many 2s present-->removes the first occurence
print(li)
#li.remove(1,2) #takes only 1 arg
#print(li)
#li.remove(10) #cannot remove an element not in list --> Error
```

```
[1, 'abc', 6.5, 1, 2, 3]
```

[27]:
```python
## pop element from a list

li = [1, 2, 'abc', 6.5, 1, 2, 3]
li.pop() #pops out the last element
print(li)
li.pop(1) #pops element from index 1
print(li)
#li.pop(10) #index is invalid -->error
#print(li)
```

```
[1, 2, 'abc', 6.5, 1, 2]
[1, 'abc', 6.5, 1, 2]
```

[1]:
```python
## Looping through List elements

#using range func
li = [1,2,'abc',3.5,7,5/2]

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

```
1
2
abc
3.5
7
2.5
```

[2]:
```python
for i in range(3,len(li)):
    print(li[i]) #accessing through indices
```

```
3.5
7
2.5
```

[4]:
```python
for element in li:
    print(element) #accessing the elements directly
```

```
1
2
abc
3.5
7
2.5
```

[5]:
```python
for element in li[2:5]: #can be sliced
    print(element) #accessing the elements directly
```

```
abc
3.5
7
```

[3]:
```
## Program: Sum of array elements

n=int(input())
arr=list(int(i) for i in input().split())
#result=0
#for number in arr:
#    result+=number
#print(result)
print(sum(arr)) #Aliter
```

```
5
2 3 4 5 6
20
```

[7]:
```
## Negative Indexing

li = [1,2,3,4,5,6]
print(li[-1]) #prints last element
print(li[-3]) #prints third last element
#print(li[-8]) #Error-->list index out of range
```

```
6
4
```

[17]:
```
## Sequencing

li = [1,2,3,4,5,6]
print(li[1:5:2]) #list(start:stop:step) from start to (stop-1) by step
print(li[:5:2]) #deafult start index 0
print(li[0::1]) #default stop is end of list
print(li[0:3:]) #default step is 1
print(li[-3:-1]) #negative indexing with sequencing
```

```
[2, 4]
[1, 3, 5]
[1, 2, 3, 4, 5, 6]
[1, 2, 3]
[4, 5]
```

[19]:
```
## Line Separated Input

N = int(input())
li = [] #initialise empty list
for i in range(N):
    element = int(input()) #conv to int else takes as ['1','2','3','4','5'] ie,␣
↪string elements
    li.append(element) #append the input element to list
print(li)
```

```
5
1
2
3
4
5
[1, 2, 3, 4, 5]
```

[23]:
```
## Space Separated Input

str = input() #gets the input as string from user by default
str_split = str.split(' ') # splits the list by (delimiter) specified, Default␣
↪is ' '
print(str_split)
print(type(str_split))

li = [] #initialize list
for element in str_split: #traverse through the string and add to new list
    li.append(int(element)) #append the converted element to the list from the␣
↪str
print(li)
print(type(li))
```

```
1 2 3 4 5
['1', '2', '3', '4', '5']
<class 'list'>
[1, 2, 3, 4, 5]
<class 'list'>
```

[25]:
```
## Space Separated Input - Aliter

li = [int(element) for element in input().split()] #converts in a single line
print(li)
```

```
1 2 3 4 5
[1, 2, 3, 4, 5]
```

[27]:
```
## Linear Search

#Print index if element found else print -1
li = [int(element) for element in input().split()] #get space separated input

n = int(input()) #find this no in list
```

```
isFound = False
for i in range(len(li)):
    if li[i] == n:
        print('index',i) #print the index
        isFound = True
        break
if not(isFound):
    print('not found', '-1')
```

```
1 2 4 5 6
8
not found -1
```

[29]:
```
## Linear Search using function

#Print index if element found else print -1
def linear_search(li,element):
    for i in range(len(li)):
        if li[i] == element:
            return i
    return -1

li = [int(element) for element in input().split()]
n = int(input()) #the no to be searched
index = linear_search(li,n)
print(index)
```

```
1 2 3 4 5
9
-1
```

0.1.1 Mutable & Immutable Concept

[34]:
```
## Immutable concept
## Value in memory not changed only reference changes
## Variables are immutable in Python

a = 3
b = 3
# Both have the same references
print('a', id(a))
print('b', id(b),'\n')
a = 4
print('a', id(a)) # References changes as the value it holds changes
print('b', id(b),'\n')
b = a
print('a', id(a)) # References changes as the value it holds changes
print('b', id(b),'\n')
```

```
a 8791220103008
b 8791220103008

a 8791220103040
b 8791220103008

a 8791220103040
b 8791220103040
```

[40]:

```
## Mutable concept
## Lists are mutable in Python
## If two lists point to SAME reference, and if val in one changes, other␣
 ↪reflects

li1 = [1,2,3,4]
print('li1',li1, id(li1))
li2 = li1 # li2 and li1 hold same reference
print('li2',li2, id(li2), '\n')

li2[0] = 5
print('li1',li1, id(li1)) # li1 also changes when li2 changed
print('li2',li2, id(li2),'\n') #both still has the same refernce

# However when the references are DIFFERENT, change in value in one does not␣
 ↪affect other
li2 = [6,7,8] #changing the reference of li2
print('li1',li1, id(li1))
print('li2',li2, id(li2),'\n') #li2 ref changed

li2[0] = 9
print('li1',li1, id(li1)) #li1 remains unchanged as the referncces are DIFFERENT
print('li2',li2, id(li2),'\n') #li2 value changed
```

```
li1 [1, 2, 3, 4] 86981312
li2 [1, 2, 3, 4] 86981312

li1 [5, 2, 3, 4] 86981312
li2 [5, 2, 3, 4] 86981312

li1 [5, 2, 3, 4] 86981312
li2 [6, 7, 8] 86907328

li1 [5, 2, 3, 4] 86981312
li2 [9, 7, 8] 86907328
```

[50]:
```
## Passing Variables through Functions

def increment(a):
    a = a + 2
    print('Inside Func', 'a', a, id(a)) #different ref
    return

a = 1
print('Outside Func', 'a', a, id(a))
increment(a)
print(a, id(a)) #value not changed
```

```
Outside Func a 1 8791220102944
Inside Func a 3 8791220103008
1 8791220102944
```

[51]:
```
def increment(a):
    a = a + 2
    print('Inside Func', 'a', a, id(a))
    return a #value returned

a = 1
print('Outside Func', 'a', a, id(a))
a = increment(a) #value updated
print(a, id(a))
```

```
Outside Func a 1 8791220102944
Inside Func a 3 8791220103008
3 8791220103008
```

[52]:
```
def increment():
    global a
    a = a + 2
    print('Inside Func', 'a', a, id(a)) #different ref created
    return

a = 1
print('Outside Func', 'a', a, id(a))
increment() #parameter need not be passed
print(a, id(a)) #value changed
```

```
Outside Func a 1 8791220102944
Inside Func a 3 8791220103008
3 8791220103008
```

[53]:
```
## Passing Lists through Functions

def increment_list(li):
    li[0] = li[0] + 2
    print('Inside Function', 'li', li, id(li)) #same reference, therefore value␣
↪changes
    return

li = [1,2,3,4]
print('Outside Function','li', li, id(li))
increment_list(li)
print(li, id(li))
```

```
Outside Function li [1, 2, 3, 4] 86956096
Inside Function li [3, 2, 3, 4] 86956096
[3, 2, 3, 4] 86956096
```

[55]:
```
def increment_list(li):
    li[0] = li[0] + 2
    print('Inside Function1', 'li', li, id(li)) #same reference, therefore␣
↪value changes
    li = [6,7,8,9]
    print('Inside Function2', 'li', li, id(li)) #li(inc) new ref created
    return

li = [1,2,3,4]
print('Outside Function','li', li, id(li))
increment_list(li)
print(li, id(li)) #prints li(main), holds the prev value not li(inc)
```

```
Outside Function li [1, 2, 3, 4] 86851008
Inside Function1 li [3, 2, 3, 4] 86851008
Inside Function2 li [6, 7, 8, 9] 83787264
[3, 2, 3, 4] 86851008
```

[57]:
```
def increment_list(li):
    li[0] = li[0] + 2
    print('Inside Function1', 'li', li, id(li)) #same reference, therefore␣
↪value changes
    li = [6,7,8,9]
    print('Inside Function2', 'li', li, id(li)) #li(inc) new ref created
    return li #li(inc) returned

li = [1,2,3,4]
print('Outside Function','li', li, id(li))
li = increment_list(li) #li(main) updated with li(inc) returned
print(li, id(li)) #prints new updated li(main)
```

```
Outside Function li [1, 2, 3, 4] 86850816
Inside Function1 li [3, 2, 3, 4] 86850816
Inside Function2 li [6, 7, 8, 9] 83613056
[6, 7, 8, 9] 83613056
```

0.1.2 Programs- Lists

[61]:
```
## Reverse a List
```