List

Built-in mutable sequence.
or
List is an **ordered** collection of elements

note lists are mutable

In [13]:

```
type([])
```

Out[13]:

```
list
```

In [7]:

```
list_1=["asit",1,234,"2233"]
```

In [10]:

```
print(type(list_1[3]))
print(type(list_1[2]))
```

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

In [19]:

```
#list function iterate over all the characters of the sttring
str_1="string"
str_2="this is a string"

list(str_1)
```

Out[19]:

```
['s', 't', 'r', 'i', 'n', 'g']
```

In [20]:

```
#return type os .split function is a list
print(str_2.split(" "))
print(type(str_2.split(" ")))
```

```
['this', 'is', 'a', 'string']
<class 'list'>
```

In [42]:

```
#Slicing of list
lst1= str_2.split(" ")

#[]
print(lst1[3])

#[:] list slicing
print(lst1[:2])
print(lst1[:]) #prints all elements
print(lst1[::-1]) #prints reverse list
print(lst1[-4::-2])
print(lst1[-2::-2])
```

```
string
['this', 'is']
['this', 'is', 'a', 'string']
['string', 'a', 'is', 'this']
['this']
['a', 'this']
```

In [ ]:

In [ ]:

In [43]:

```
## concatenation operation
lst1+["concatination",5]
```

Out[43]:

```
['this', 'is', 'a', 'string', 'concatination', 5]
```

In [44]:

```
#concatination of nested list
## concatenation operation
lst2=lst1 + [['new element',3]]
print(lst2)
```

```
['this', 'is', 'a', 'string', ['new element', 3]]
```

In [45]:

```
#Arithmatic operation on a list
lst1*2
```

Out[45]:

```
['this', 'is', 'a', 'string', 'this', 'is', 'a', 'string']
```

In [48]:

```
#mutability of list
lst2[4]="manipulation"
print(lst2)
```

```
['this', 'is', 'a', 'string', 'manipulation']
```

In [54]:

```
#finding element in a string
if "string" in lst1:
    print("preent")
#prefer this solution
```

```
preent
```

In [55]:

```
for i in lst2:
    if i=="string":
        print("present")
        break
```

```
present
```

note avoid iteration by using if statement

In [ ]:

In [ ]:

In [56]:

```
## check elements inside a list
"string"in lst2
```

Out[56]:

```
True
```

In [57]:

```
2.0 in lst2
```

Out[57]:

```
False
```

In [69]:

```
1.9999999999999999 in [2, 4, 5] #bugg in python
```

Out[69]:

```
True
```

In [70]:

```
5==5.0
```

Out[70]:

```
True
```

In [88]:

```
# Maximun and Minimum
lst1=["Zebra","Monkey","Donkey","Lion"]
lst2=[5,6,2,9,5,8,6]
```

In [89]:

```
print(max(lst1)) #print maximum based on ascii value
print(max(lst2))
print(min(lst1))
print(min(lst2))
```

```
Zebra
9
Donkey
2
```

In [90]:

```
#Apprend
print(lst1)
lst1.append(["list",123])
print(lst1)
```

```
['Zebra', 'Monkey', 'Donkey', 'Lion']
['Zebra', 'Monkey', 'Donkey', 'Lion', ['list', 123]]
```

[note]:- **.append** is an inplace operation with which the the list gets updated in the orignal variable so------ ***lst_new = lst1.append("a")*** **is not a correct operation**

In [91]:

```
lst_new = lst1.append("a")
print(lst_new)
```

```
None
```

.extend() --use this function to concatinate to string cuz if we use .append() its just add the list as is makiing the list a nested one.

In [102]:

```
lst1=["Zebra","Monkey","Donkey","Lion"]
lst2=[5,6,2,9,5,8,6]
lst1.append(lst2)
print(lst1)

lst1=["Zebra","Monkey","Donkey","Lion"]
lst2=[5,6,2,9,5,8,6]
lst1.extend(lst2)
print(lst1)
```

```
['Zebra', 'Monkey', 'Donkey', 'Lion', [5, 6, 2, 9, 5, 8, 6]]
['Zebra', 'Monkey', 'Donkey', 'Lion', 5, 6, 2, 9, 5, 8, 6]
```

In [92]:

```
#removal of element ,pop()
print(lst1)
lst1.pop() #removes the last element
print(lst1)
```

```
['Zebra', 'Monkey', 'Donkey', 'Lion', ['list', 123], 'a']
['Zebra', 'Monkey', 'Donkey', 'Lion', ['list', 123]]
```

In [93]:

```
print(lst1)
lst1.pop(2) #removes the 2nd element
print(lst1)
```

```
['Zebra', 'Monkey', 'Donkey', 'Lion', ['list', 123]]
['Zebra', 'Monkey', 'Lion', ['list', 123]]
```

[note]:- **.pop()** is an inplace operation but it also returns the removed element as value.

In [94]:

```
print(lst1)
a=lst1.pop(2) #removes the 2nd element
print(lst1)
print(a)
```

```
['Zebra', 'Monkey', 'Lion', ['list', 123]]
['Zebra', 'Monkey', ['list', 123]]
Lion
```

In [ ]:

```

```

In [103]:

```
## Sorting and Reverse method in list
new_list=['q','e','f','s','t','u']
print(new_list)

print(new_list[::-1])

new_list.reverse()
print(new_list)

new_list.sort()
print(new_list)

#note both .reverse and .sort are inplace functions
```

```
['q', 'e', 'f', 's', 't', 'u']
['u', 't', 's', 'f', 'e', 'q']
['u', 't', 's', 'f', 'e', 'q']
['e', 'f', 'q', 's', 't', 'u']
```

In [105]:

```
#sorting in descending order
new_list.sort(reverse=True) #press shift +tab to see the arguments and it says by default
print(new_list)
```

```
['u', 't', 's', 'q', 'f', 'e']
```

In [ ]:

```
```

In [ ]:

```
```

Nested List

In [106]:

```
# Let's make three lists
lst_1=[1,2,3]
lst_2=[4,5,6]
lst_3=[7,8,9]

# Make a list of lists to form a matrix
matrix = [lst_1,lst_2,lst_3]
```

In [107]:

```
matrix
```

Out[107]:

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

In [115]:

```
#selecting elements from a matrix

#extract 4 from matrix
print(matrix[1][0])
print(type(matrix[1][0]))


#extract elements 7 and 8
print(matrix[2][:2])

#extract 1,4,7
#not possible here it can be done in arrays by using numpy
```

```
4
<class 'int'>
[7, 8]
```

```
```

```
```

In [116]:

```
## List Comprehension
[i for i in range(20)]
```

Out[116]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [121]:

```
## List Comprehension
## Even numbers
[i if i%2==0 else "ODD" for i in range(0,20)]
```

Out[121]:

```
[0,
 'ODD',
 2,
 'ODD',
 4,
 'ODD',
 6,
 'ODD',
 8,
 'ODD',
 10,
 'ODD',
 12,
 'ODD',
 14,
 'ODD',
 16,
 'ODD',
 18,
 'ODD']
```

In [123]:

```
[i for i in range(20) if i%2==0] #list of even nos. using list comprehension
```

Out[123]:

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [126]:

```
## Assignment
## Sum of even numbers and odd numbers
lst=[1,2,3,4,5,6,7,8]

#solution
even_sum=0
odd_sum=0
for i in lst:
    if i%2==0:
        even_sum+=i
    else:
        odd_sum+=i
print(even_sum)
print(odd_sum)

#or
#using list comprehension
even_sum2=sum([num for num in lst if num%2==0])
print(even_sum2)

odd_sum2=sum([num for num in lst if num%2!=0])
print(odd_sum2)
```

```
20
16
20
16
```

In [129]:

```
#using list comprehension find the square of all the nos in the list
lst=[1,2,3,4,5,6,7,8,9,10]
[num**2 for num in lst]
```

Out[129]:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [130]:

```
# Example 2: Create a list of only the positive numbers from a given list\
numbers = [-2, -1, 0, 1, 2, 3, 4]
[num for num in numbers if num>0]
```

Out[130]:

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

In [135]:

```
 # Example 3: Create a list of only the first letters of words in a list
words = ['apple', 'banana', 'cherry', 'date']
[i[0] for i in words]
```

Out[135]:

```
['a', 'b', 'c', 'd']
```

In [136]:

```
# Example 4: Convert a list of temperatures from Celsius to Fahrenheit uing list comprehe
celsius_temperatures = [0, 10, 20, 30, 40, 50]
#concept-- (9/5)*temp+32

[(9/5)*i+32 for i in celsius_temperatures]
```

Out[136]:

```
[32.0, 50.0, 68.0, 86.0, 104.0, 122.0]
```

In [140]:

```
# Example 5: Flatten a list of lists into a single list
lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

[j for i in lists for j in i]
```

Out[140]:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [187]:

```
## Assignment
## Using both code and list comprehesnion
# Example 2: Create a list of only the prime numbers from a given list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[x for x in numbers if all(x % y != 0 for y in range(2,x))]
```

Out[187]:

```
[1, 2, 3, 5, 7]
```

In [188]:

```
[i for i in numbers for j in range(1,i) if i%j==0]
```

Out[188]:

```
[2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 8, 8, 9, 9, 10, 10, 10]
```

In [189]:

```
# Example 3: Create a list of all the possible combinations of 2 elements from a list
numbers = [1, 2, 3, 4, 5]
```

In [ ]:

```
```