Module02_Day02_PyRefresher_2

December 16, 2022

1 Python Refresher 2

Underscore

```
[ ]: _ = 56
```

```
[ ]: _
```

```
[ ]: 56
```

```
[ ]: del _
```

```
[ ]: 6 + 2
```

```
[ ]: 8
```

Restarted Kernel here

```
[ ]: 6 + 2
```

```
[ ]: 8
```

```
[ ]: _ # it saves previous not stored value
```

```
[ ]: 8
```

```
[ ]: _ * _
```

```
[ ]: 64
```

```
[ ]: "Hello" + "World"
```

```
[ ]: 'HelloWorld'
```

```
[ ]: _
```

```
[ ]: 'HelloWorld'
```

1.1 Built in Data Structures

Why Data Structures are important?: To store and access different types of data efficiently.

1.1.1 List

- Capable of storing multiple terms
- Heterogenous Values: means can store differnt data types
- Mutable
- Not mapped to continuous memory locations

[ ]:
```
# Empty List

marks = list()

print(marks,type(marks))
```

```
[] <class 'list'>
```

[ ]:
```
# Lists are hetrogenous

marks = [92,100,44,59,"ab"]
marks
```

[ ]:
```
[92, 100, 44, 59, 'ab']
```

[ ]:
```
# indexing

marks[0]
```

[ ]:
```
92
```

[ ]:
```
# Length of list
len(marks)
```

[ ]:
```
5
```

[ ]:
```
marks[len(marks) - 1]
```

[ ]:
```
'ab'
```

[ ]:
```
# Can be shortend as
marks[-1]
```

[ ]:
```
'ab'
```

Slicing

[ ]:
```
# Slicing

marks = [67,92,99,56,44,71,87,90,94,100,66]

marks[2:5]
```

```
[ ]: [99, 56, 44]
```

```
[ ]: marks[:5]
```

```
[ ]: [67, 92, 99, 56, 44]
```

```
[ ]: marks[-1:-4]
```

```
[ ]: []
```

```
[ ]: marks[-3:]
```

```
[ ]: [94, 100, 66]
```

```
[ ]: marks[:]
```

```
[ ]: [67, 92, 99, 56, 44, 71, 87, 90, 94, 100, 66]
```

```
[ ]: _
```

```
[ ]: [67, 92, 99, 56, 44, 71, 87, 90, 94, 100, 66]
```

```
[ ]: marks[::2]
```

```
[ ]: [67, 99, 44, 87, 94, 66]
```

```
[ ]: marks[::3]
```

```
[ ]: [67, 56, 87, 100]
```

```
[ ]: marks[9:6:-1]
```

```
[ ]: [100, 94, 90]
```

```
[ ]: # Python is smart to figure it out it starts from negative
     marks[::-1]
```

```
[ ]: [66, 100, 94, 90, 87, 71, 44, 56, 99, 92, 67]
```

```
[ ]: id(marks[0]),id(marks[1])
```

```
[ ]: (1552503755184, 1552503755984)
```

Mutability of list

```
[ ]: # Individual elements ids will change, but list won't
```

```
[ ]: marks = [20,30,50,80]
     marks
```

```
[ ]: [20, 30, 50, 80]
```

```
[ ]: marks[0] = 90
```

```
[ ]: id(marks)
```

```
[ ]: 1552622769600
```

```
[ ]: marks[0] = marks[0] + 1
```

```
[ ]: id(marks)
```

```
[ ]: 1552622769600
```

List Methods

```
[ ]: marks.append(100)
     marks, id(marks)
```

```
[ ]: ([92, 30, 50, 80, 100], 1552622769600)
```

```
[ ]: id(marks.copy())
```

```
[ ]: 1552587907392
```

```
[ ]: marks.insert(2,40) # insert 40 before index 2
     marks
```

```
[ ]: [92, 30, 40, 40, 50, 80, 100]
```

```
[ ]: marks.pop() # remove and return last element
```

```
[ ]: 100
```

```
[ ]: marks.count(40)
```

```
[ ]: 1
```

```
[ ]: marks.reverse()
     marks
```

```
[ ]: [80, 50, 40, 30, 92]
```

```
[ ]: marks.sort(reverse=True)
```

```
[ ]: marks
```

```
[ ]: [92, 80, 50, 40, 30]
[ ]: random = [67,45,10,"ab","alpha"] # Cannot compare string and integer
     random.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5180\1405085579.py in <module>
      1 random = [67,45,10,"ab","alpha"]
----> 2 random.sort()

TypeError: '<' not supported between instances of 'str' and 'int'
```

```
[ ]: lst = [4,6,7]
     lst.append([[1],2,3])
```

```
[ ]: lst
```

```
[ ]: [4, 6, 7, [[1], 2, 3]]
```

```
[ ]: lst.extend([2,3])
```

```
[ ]: lst
```

```
[ ]: [4, 6, 7, [[1], 2, 3], 2, 3]
```

```
[ ]: lst.extend([[2,3]])
```

```
[ ]: lst
```

```
[ ]: [4, 6, 7, [[1], 2, 3], 2, 3, [2, 3]]
```

```
[ ]: lst.extend([1]) # works because [1] is iterable
```

```
[ ]: lst
```

```
[ ]: [4, 6, 7, [[1], 2, 3], 2, 3, [2, 3], 1]
```

```
[ ]: lst.extend(1) # 1 is not iterable as mentioned in docstring
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5180\780194928.py in <module>
----> 1 lst.extend(1)

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

```
[ ]: [1,2,3] + [4,5] # + Works like extend
[ ]: [1, 2, 3, 4, 5]
```

```
[ ]: [1,2,3] + 5 # Here it gives error because it can concatenate only lists
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5180\1834913151.py in <module>
----> 1 [1,2,3] + 5

TypeError: can only concatenate list (not "int") to list
```

```
[ ]: [1,2,3] * 3 # This concatenate
```

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

```
[ ]: [1,2,3] * [1,2] # This is invalid
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5180\577154900.py in <module>
----> 1 [1,2,3] * [1,2] # This is invalid

TypeError: can't multiply sequence by non-int of type 'list'
```

Iterate List

```
[ ]: new_marks = []
     for ele in marks:
     #     print(ele*2)
         new_marks.append(ele*2)
```

```
[ ]: new_marks, marks
```

```
[ ]: ([184, 160, 100, 80, 60], [92, 80, 50, 40, 30])
```

List Comprehension

```
[ ]: new_marks = [ele * 2 for ele in marks]
     new_marks
```

```
[ ]: [184, 160, 100, 80, 60]
```

If

```
[ ]: marks = [66, 100, 94, 90, 87, 71, 44, 56, 99, 92, 67]
     sq_marks = [ele*2  for ele in marks if ele%2==0 ]
     sq_marks
```

```
[ ]: [132, 200, 188, 180, 88, 112, 184]
```

```
[ ]: marks = [66, 100, 94, 90, 87, 71, 44, 56, 99, 92, 67]
     sq_marks = [ele*2 if ele%2==0 else ele * 3 for ele in marks ]
     sq_marks
```

```
[ ]: [132, 200, 188, 180, 261, 213, 88, 112, 297, 184, 201]
```

```
[ ]: # Better way
     def action(n):
         if n%2==0: return n * 2
         else: return n * 3

     new_marks = [action(ele) for ele in marks]
     print(new_marks)
```

```
[132, 200, 188, 180, 261, 213, 88, 112, 297, 184, 201]
```

1.1.2 Nested Lists

```
[ ]: mat = [[1,2],[3,4]] # has 2 elements, (2 lists)
```

```
[ ]: type(mat), len(mat), print(mat)
```

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

```
[ ]: (list, 2, None)
```

```
[ ]: for ele in mat:
         print(ele)
```

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

```
[ ]: mat[1]
```

```
[ ]: [3, 4]
```

```
[ ]: mat[1][1]
```

```
[ ]: 4
```

```
[ ]: # Matrix: where you have rows & columns
```

```
[ ]: mat = [[1,2,3],[4,8,9],[5,6,7],[0,1,1]]
     mat
```

```
[ ]: [[1, 2, 3], [4, 8, 9], [5, 6, 7], [0, 1, 1]]
```

[ ]:
```
for ele in mat:
    print(ele)
```

```
[1, 2, 3]
[4, 8, 9]
[5, 6, 7]
[0, 1, 1]
```

[ ]:
```
mat[1][0]
```

[ ]:
```
4
```

[ ]:
```
C = [[0]*2]*2 # Not recommended
C
```

[ ]:
```
[[0, 0], [0, 0]]
```

[ ]:
```
#because of refrencing
C[1][0] = 1
C # all columns are changed
```

[ ]:
```
[[1, 0], [1, 0]]
```

Empty 2d Matrix

[ ]:
```
D = [[0 for i in range(3)] for j in range(3)]
D
```

[ ]:
```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

[ ]:
```
A = [[1,2],[3,4]]
B = [[4,1],[0,8]]
C = [[0,0] for i in range(len(A))]


for i in range(len(A)):
    for j in range(len(A[0])):
        C[i][j] = (A[i][j] + B[i][j])

C
```

[ ]:
```
[[5, 3], [3, 12]]
```

[ ]:
```
players = ['Jadeja','Rahul','Rohit']
players[len(players):] = ["Dhoni","Virat"]
players
```

[ ]:
```
['Jadeja', 'Rahul', 'Rohit', 'Dhoni', 'Virat']
```

```
[ ]: players[-1] =["D","V"]
     players
```

```
[ ]: ['Jadeja', 'Rahul', 'Rohit', 'D', 'D', 'V', 'D', ['D', 'V']]
```

```
[ ]: a=["data", "python", "scaler", "ML", "foo", "Jupyter", "lists"]
```

```
[ ]: a[-7]
```

```
[ ]: 'data'
```

```
[ ]: 6**2
```

```
[ ]: 36
```

```
[ ]: n = 5
     sum_even = sum([i for i in range (0,n+1,2)])
     sum_odd = sum([i for i in range (1,n+1,2)])
     sum_even, sum_odd
```

```
[ ]: (6, 9)
```