DSML Intermediate : Python Refresher - 2

Recap

```
print('Hello world')
```

```
Hello world
```

```
print(8 // 3)
```

```
2
```

```
print(8 / 3)
```

```
2.6666666666666665
```

```
print(-8//3)
```

```
-3
```

```
print(-8/3)
```

```
-2.6666666666666665
```

```
print(8//-3)
```

```
-3
```

```
print(-(8//3))
```

```
-2
```

```
a = 60
if a > 50:
    print("FIRST")
if a > 40:
    print("SECOND")
```

```
FIRST
SECOND
```

```
a = 60
if a > 50:
    print("FIRST")
elif a > 40:
    print("SECOND")
```

```
FIRST
```

Data Structures

Lists

```
a = ["akash", 5, True, print, 5.67, [1, 2, 3]]
```

```
type(a)
```

```
list
```

```
len(a)
```

```
6
```

```
a[0] # 0th index
```

```
'akash'
```

```
a[1] # 1st index
```

```
5
```

```
a[-1]
```

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

```
a[-1][0]
```

```
1
```

```
a[-1][1]
```

```
2
```

```
l = [5, 1, 2, 3, 7, 8]
```

```
sum(l)
```

```
26
```

```
max(l)
```

```
8
```

```
min(l)
```

```
1
```

List Slicing

```
a = [1,2,3,4,5,6,7,8,9,10]

a[3:8:-1]
```

```
l = [5, 1, 2, 3, 7, 8]

res = l[0:3:1]
print(res)
print(id(res))
```

```
[5, 1, 2]
140477777661760
```

```
l
print(id(l))
```

```
140477777771328
```

```
res = l[2:5] # default jump, inc = 1
print(res)
```

```
[2, 3, 7]
```

```
res = l[:5] # default start = 0, default jump = 1
print(res)
```

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

```
res = l[:] # default start = 0, end = len(l), jump = 1
print(res)
```

```
[5, 1, 2, 3, 7, 8]
```

```
l[::2]
```

```
[5, 2, 7]
```

```
print(l)
```

```
[5, 1, 2, 3, 7, 8]
```

```
l[::15]
```

```
[5]
```

Negative Indexing

```
numbers = [5, 1, 2, 3, 7, 8]*2
```

```
print(numbers)
```

```
[5, 1, 2, 3, 7, 8, 5, 1, 2, 3, 7, 8]
```

```
numbers[-1:-5:-1]
```

```
[8, 7, 3, 2]
```

```
numbers[-1::-1] # negative increment, default end = -len(l) - 1 (till the start)
```

```
[8, 7, 3, 2, 1, 5, 8, 7, 3, 2, 1, 5]
```

```
numbers[::-1] # default start = -1, default end = -len(l) - 1, inc = -1
```

```
[8, 7, 3, 2, 1, 5, 8, 7, 3, 2, 1, 5]
```

```
numbers[-1:-12:-1]
```

```
[8, 7, 3, 2, 1, 5, 8, 7, 3, 2, 1]
```

```

```

Quizzes

```
numbers = [5, 1, 2, 3, 7, 8]*2


print(numbers)
```

```
[5, 1, 2, 3, 7, 8, 5, 1, 2, 3, 7, 8]
```

```
numbers[-1::]
```

```
[8]
```

```
numbers = [5, 1, 2, 3, 7, 8]
```

```
numbers[3:5:-1]
```

```
[]
```

Mix negative and postive index

```
numbers = [5, 1, 2, 3, 7, 8]
```

```
numbers[-2:4:-1]
```

```
[]
```

```
numbers[-2:5:-1]
```

```
[]
```

```
numbers[-2:2:-1]
```

```
[7, 3]
```

```
numbers = [5, 1, 2, 3, 7, 8, 5, 1, 2, 3, 7, 8]
```

```
# see the index as a position not number
numbers[5:-3:1]
```

```
[8, 5, 1, 2]
```

Iterate on Lists

```
numbers = [5, 1, 2, 3, 7, 8, 5, 1, 2, 3, 7, 8]

for i in numbers:
    print(i, end=' ')
```

```
5 1 2 3 7 8 5 1 2 3 7 8
```

Mutability

```
bank_accounts_used_by_me = [3000, 0, 5000]
```

```
bank_accounts_used_by_dad = bank_accounts_used_by_me
```

```
bank_accounts_used_by_me[0] -= 1000
```

```
print(bank_accounts_used_by_me)
```

```
[2000, 0, 5000]
```

```
print(bank_accounts_used_by_dad)
```

```
[2000, 0, 5000]
```

What?

```
l1 = [3000, 0, 5000] # me
l2 = l1 # dad

l1[0] -= 1000 # I used money
print(l1) # my copy changes
```

```
[2000, 0, 5000]
```

```
print(l2)
```

```
[2000, 0, 5000]
```

Shallow Copy

```
# same object being used
```

```
l1 = [3, 4, 1, 2, 5]

l2 = l1
print(id(l1) == id(l2))
```

```
True
```

```
l1[0] = 7 # lists are mutable, changing the same object


print(id(l1) == id(l2))
```

```
True
```

```
print(l1)
```

```
[7, 4, 1, 2, 5]
```

```
print(l2)
```

```
[7, 4, 1, 2, 5]
```

```
a = 1234
b = a

print(id(a) == id(b))
```

```
True
```

```
a = 1234
b = a
b = 3 # immutable, creating a new object
print(id(a) == id(b))

print(a)
print(b)
```

```
False
1234
3
```

```
a = 1234
b = a
a = 3
print(id(a) == id(b))
print(a)
print(b)
```

```
False
3
1234
```

MCQ

```
my_teams = ['Raptors', 'Heat', 'Nets']
your_teams = my_teams
print(id(my_teams) == id(your_teams))
```

```
True
```

```
my_teams[1] = 'Lakers'
print(id(my_teams) == id(your_teams))
```

```
True
```

```
print('My teams are:', my_teams)
print('Your teams are:', your_teams)
```

```
My teams are: ['Raptors', 'Lakers', 'Nets']
Your teams are: ['Raptors', 'Lakers', 'Nets']
```

HW

```
board = [["", "", ""], ["", "", ""], ["", "", ""]]
```

```
board[0][0] = '0'
```

```
print(board)
```

```
[['0', '', ''], ['', '', ''], ['', '', '']]
```

```

```

```
board = [""]*3
```

```
print(board)
```

```
['', '', '']
```

```
# ['', '', ''] * 3
```

```
[[''] * 3]
```

```
[['', '', '']]
```

Magic

```
board = [[''] * 3] * 3
```

```
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

```
board[0][0] = '0'
```

```
print(board)
```

```
[['0', '', ''], ['0', '', ''], ['0', '', '']]
```

```
board[1][1] = 'X'
```

```
print(board)
```

```
[['0', 'X', ''], ['0', 'X', ''], ['0', 'X', '']]
```

Hint: Think about mutability and immutability

Strings => Immutable

Lists => Mutable

List Multiplication

Up Next

0. HW to be discussed
1. Tuples
2. Sets and Dictionaries
3. Lists => Deep Copy
4. Comprehension
5. Some more important methods in the list, set, tuples, dict

Doubts

```
L = [10, 20, 30, 40, 50, 60]
print(L[2:4:15])
```

```
[30]
```

```
print(L[2:4:-1])
```

```
[]
```

```
print(L[-1:4:-1])
```

```
[60]
```

```
print(L[-3:-1:-1])
```

```
[]
```

```
print(L[-3:-1:1])
```

```
[40, 50]
```

```
L = [10, 20, 30, 40, 50, 60]


L[0::-1]
L[0::-2]
```

```
[10]
```

```
L[0:0:-1]
```

```
[]
```

```
L[0:(-len(L) - 1):-1]
```

```
[10]
```

```
L[0::1]
```

```
[10, 20, 30, 40, 50, 60]
```

```
L[-1::]*2
```

```
[60, 60]
```

```
[60]*2
```

```
[60, 60]
```

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

```
max(l)
```

```
[7, 8, 9]
```

```
min(l)
```

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

```
a = [1,2,3,4]
b = a
c = [4,5,6]
a.append(c)
# a = a.append(c) -> append method returns None
```

```
print(a)
```

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

```
print(b)
```

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

```
a=1234
b=1234
id(a)==id(b)
```

```
False
```

```
a = 1234
b = a
id(a) == id(b)
```

```
True
```

```
a = 257
b = 257
print(id(a) == id(b))
```

```
False
```

```
# small integer caching
a = 256
b = 256
print(id(a) == id(b))
```

```
True
```

```
numbers = [5, 1, 2, 3, 7, 8]
numbers[6] = [100]
```

```
---------------------------------------------------------------------------


IndexError                                Traceback (most recent call last)


Input In [169], in <cell line: 2>()
      1 numbers = [5, 1, 2, 3, 7, 8]
----> 2 numbers[6] = [100]



IndexError: list assignment index out of range
```

```
# add the data to the list
numbers[6:] = [100]
numbers
```

```
[5, 1, 2, 3, 7, 8, 100]
```

```
numbers = [1, 2, 3, 4, 5, 7]
```

```
numbers = [i*2 for i in numbers] # comprehension => in next class
```

```
numbers
```

```
[4, 8, 12, 16, 20, 28]
```