DSML Intermediate : Python Refresher - 4

Recap

How to solve a problem?

1. **Before implementing the code =>**

- try on paper,
- take examples,
- think the logic,
- plan the solution

2. **Write the code, test it (dry run on an example) and submit it**

3. **You will get errors, how to get past those hurdles?**

Before reaching out to TAs:

- If the problem has a video solution or a hint 1 / hint 2 => try to use it
- If it's a syntax error => try to google, stackoverflow => finding / debugging code is a process which develops over time

Then still if not resolved, reach out to TAs / peers in slack / whatsapp

If it's a MCQ problem

- try running the code in IDE and figuring out what is happening

4. **Say you finally solved problem with helps of peers / TAs what next?**

- Don't jump to the next problem yet!!!!
- Checkout the hints (they are now free - no score deduction)
- Maybe your post solution on slack / whatsapp group and get feedback from peers
- reach out TAs to know how can you have made your own solution much better

5. **(Optional - for revision)**

- bookmark a set of problems which you found out slightly challenging (where you learnt something new)
- make your own notes of what you learnt while solving a problem

Goal =>

1. PSP => 70+
2. PSP => 80+
3. PSP => 85+
4. PSP => 90+

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

print(board)
```

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

```
print(id(board[0][0]) == id(board[1][1]) == id(board[2][2]))
```

```
True
```

```
tup1 = (1,2)
tup2 = (1,2)

print(id(tup1) == id(tup2))
```

```
False
```

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

print(id(l1[0]))
print(id(l2[0]))
```

```
140474019014960
140474019014960
```

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

```
140474305431168
140474305243136
```

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

```
False
```

Comprehension

```
fruits = ['apples', 'bananas', 'strawberries', 'grapes',
          'mango', 'oranges', 'cherry']
```

```
for i in fruits:
    print(i)
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

```
[print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry




[None, None, None, None, None, None, None]
```

```
x = print(5)
```

```
5
```

```
print(x)
```

```
None
```

```
res = [print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

```
print(res)
```

```
[None, None, None, None, None, None, None]
```

Squares of all elements

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

```
b = []

for i in a:
    b.append(i*i)
```

```
print(b)
```

```
[25, 1, 4, 9, 49]
```

```
b_cool = [i**2 for i in a]
```

```
print(b_cool)
```

```
[25, 1, 4, 9, 49]
```

```
c = [i for i in a]
```

```
print(c)
```

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

Quiz

```
a = [1, 2, 3]

l = [0 for i in a]
```

```
print(l)
```

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

```
print(sum(l))
```

```
0
```

With condition

```
a = [5, 1, 2, 3, 7, 6]

b = []

for i in a:
    if i % 2 == 0: # divisible by 2 => even
        b.append(i*i)
    else:
        b.append(i)

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

```
def my_logic(i):
    if i % 2 == 0:
        return i*i
    else:
        return i
```

```
my_logic(5)
```

```
5
```

```
b = []

for i in a:
    b.append(my_logic(i))

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

```
b = [my_logic(i) for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

```
b = [i*i if i % 2 == 0 else i for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

```
i = 5

if i == 5:
    print('True')
else:
    print('False')
```

```
True
```

```
i = 5
'True' if i == 5 else 'False'
```

```
'True'
```

```
def my_print(i):
    print(i)
    return None
```

```
res = [my_print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

```
print(res)
```

```
[None, None, None, None, None, None, None]
```

Quizzes

```
fruits = ["apples", "bananas", "strawberries"]
fruits = [fruit.upper() for fruit in fruits]
```

```
print(fruits)
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```

```
s = "Rahul"
```

```
s2 = s.upper()
print(s2)
```

```
RAHUL
```

```
print(s)
```

```
Rahul
```

```
bits = [False, True, False, False, True, False, False, True]
bits = [1 if b==True else 0 for b in bits]
```

```
print(bits)
```

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

More fun on comprehension

```
l = [i for i in range(1, 5)]
print(l)
```

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

```
s = { i for i in range(1, 10) }
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
t = ( i for i in range(1, 5) )
print(t)
```

```
<generator object <genexpr> at 0x7fc2b90727b0>
```

```
tuple(t)
```

```
(1, 2, 3, 4)
```

```
d = { i : i**2 for i in range(1, 5)}
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

```
tuple([1, 2, 3, 4])
```

```
(1, 2, 3, 4)
```

```
d = { i : 5 for i in range(1, 3)}
print(d)
```

```
{1: 5, 2: 5}
```

```
d = { 5 : i for i in range(1, 3)}
print(d)
```

```
{5: 2}
```

```
d = { 5 : i for i in range(10, 2, -1)}
print(d)
```

```
{5: 3}
```

Fun part of list comprehension

```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
```

```
print(l)
```

```
[0, 0, 0, 1, 0, 2]
```

```
res = [i for i in range(3)]

print(res)
```

```
[0, 1, 2]
```

```
res2 = [j for j in range(2)]
print(res2)
```

```
[0, 1]
```

```
cool_res = [i*j for j in range(2) for i in range(3)]

print(cool_res)
```

```
[0, 0, 0, 0, 1, 2]
```

```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
print(l)

# single comprehension
res_cool = [i*j for i in range(3) for j in range(2)]
print(res_cool)
```

```
[0, 0, 0, 1, 0, 2]
[0, 0, 0, 1, 0, 2]
```

2d list

```
a = []
for i in range(3):
    b = []
    for j in range(2):
        b.append(i * j)
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

```
[0*j for j in range(2)]
```

```
[0, 0]
```

```
[1*j for j in range(2)]
```

```
[0, 1]
```

```
[2*j for j in range(2)]
```

```
[0, 2]
```

```
# comprehension within comprehension

super_cool = [[i*j for j in range(2)] for i in range(3)]
```

```
print(super_cool)
```

```
[[0, 0], [0, 1], [0, 2]]
```

```
# multiplication tables of all numbers from 1 to 7
```

```
res = []
for i in range(1, 8):
    c = []
    for j in range(1, 11):
        c.append(i * j)
    res.append(c)

for i in res:
    print(i)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
```

```
multiplication_table = [[i*j for j in range(1, 11)] for i in range(1, 8)]

for i in multiplication_table:
    print(i)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
```

Dictionaries

```
person = {
    "name": "Numan",
    "age": 5000,
    "greatest_avenger": True
}
```

```
person['name']
```

```
'Numan'
```

```
# add a new key or update value at a key

person['weapons'] = ["mjolnir", "stormbreaker", "good looks"]
```

```
print(person)
```

```
{'name': 'Numan', 'age': 5000, 'greatest_avenger': True, 'weapons': ['mjolnir', 'stormbreaker', 'good looks']}
```

```
person['weapons'].pop()
```

```
'good looks'
```

```
print(person['weapons'])
```

```
['mjolnir', 'stormbreaker']
```

```
# update many keys or add new keys

person.update({
    'age': 100000,
    'weapons': ['stormbreaker'],
    'random_info': ("random", "info")
})
```

```
print(person)
```

```
{'name': 'Numan', 'age': 100000, 'greatest_avenger': True, 'weapons': ['stormbreaker'], 'radom_info': ('random', 'info'), 'random_in
```

```
person[0] # only keys can be used as an index
```

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

KeyError                                  Traceback (most recent call last)

Input In [167], in <cell line: 1>()
----> 1 person[0]


KeyError: 0
```

```
person.update({
    [1, 2, 3]: "list"
})
```

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

TypeError                                 Traceback (most recent call last)

Input In [168], in <cell line: 1>()
----> 1 person.update({
      2     [1, 2, 3]: "list"
      3 })


TypeError: unhashable type: 'list'
```

```
person.update({
    (1, 2, 3): 'valid value'
})
```

```
print(person)
```

```
{'name': 'Numan', 'age': 100000, 'greatest_avenger': True, 'weapons': ['stormbreaker'], 'radom_info': ('random', 'info'), 'random_in
```

```
avengers = [
    'bruce',
    {
        'foo': 1,
        'bar':
        {
            'z' : 30
        },
        'baz': 3
    }
]
```

```
avengers[1]['bar']['z']
```

```
30
```

```
avengers[1]['bar']
```

```
{'z': 30}
```

```
# avengers[1]['bar']['a']

avengers[1]['bar'].get('a', 0)
```

```
0
```

```
avengers[1]['bar'].get('z')
```

```
30
```

```
avengers[1]['bar'].get('a', 0)
```

```
0
```

```

```

```

```

Sets

```
a = {5, 1, 2, 4, 5, 5, 2, 4, 5, 6, 67}
```

```
print(a)
```

```
{1, 2, 67, 4, 5, 6}
```

```
s = { (1, 2), (1, 2), (1, 3)}

print(len(s))
```

```
2
```

```
print(s)
```

```
{(1, 2), (1, 3)}
```

```
(1, 2) == (1, 2)
```

```
True
```

```
{1, 2, 3} == {3, 2, 1, 3, 2, 3, 1, 2}
```

```
True
```

```

```

```
s = { [1, 2], [1, 2]}
```

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

TypeError                                 Traceback (most recent call last)

Input In [193], in <cell line: 1>()
----> 1 s = { [1, 2], [1, 2]}


TypeError: unhashable type: 'list'
```

```

```

```
# empty set
s = set()
s.add(7)
s.add(7)

s.add((1, 2))
s.add("hello")
s.add("HELLO")

print(s)

# sets are unordered
```

```
{'HELLO', (1, 2), 'hello', 7}
```

Set Theory

```
s1 = {1, 2, 3, 7}

s2 = {3, 7, 4, 5}
```

```
res = s1 | s2 # union
print(res)
```

```
{1, 2, 3, 4, 5, 7}
```

```
res = s1.union(s2) # union
print(res)
```

```
{1, 2, 3, 4, 5, 7}
```

```
res = s1 & s2 # intersection
print(res)
```

```
{3, 7}
```

```
res = s1 - s2 # difference, in s1 but not in s2
print(res)
```

```
{1, 2}
```

```
res = s2 - s1 # difference, in s2 but not in s1
print(res)
```

```
{4, 5}
```

```
res = s1 ^ s2 # xor operator, symmetric difference
print(res)
```

```
{1, 2, 4, 5}
```

```
res = s1.symmetric_difference(s2)
print(res)
```

```
{1, 2, 4, 5}
```

```
s
```

```
{(1, 2), 7, 'HELLO', 'hello'}
```

```
s.remove((1, 2))
```

```
print(s)
```

```
{'HELLO', 'hello', 7}
```

```

```

```

```

Challenge Problems

```

```

Doubts

```
tup = ("Orange", [10, 20, 30], (5, 15, 25))

tup[1:2]
```

```
([10, 20, 30],)
```

```
tup[1:2][0]
```

```
[10, 20, 30]
```

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

```
30
```

```
l1 = [100001214, 5, 7]
l2 = [100001214, 5, 7]
```

```
id(l1[0]) == id(l2[0])
```

```
False
```

```
l1 = [1, 5, 7]
l2 = [1, 5, 7]

id(l1[0]) == id(l2[0])
```

```
True
```

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

```
False
```

```
l1 = [[1, 5, 7]] * 2
print(l1)
```

```
[[1, 5, 7], [1, 5, 7]]
```

```
id(l1[0]) == id(l1[1])
```

```
True
```

```
l2 = [[1, 5, 7], [1, 5, 7]]
print(l2)
```

```
[[1, 5, 7], [1, 5, 7]]
```

```
id(l2[0]) == id(l2[1])
```

```
False
```

```

```

```

```

```

```

```
t1 = (1, 2)

t2 = (1, 2)
```

```
id(t1) == id(t2)
```

```
False
```

```
id(t1[0]) == id(t2[0])
```

```
True
```

```
t1 = (19491241841, 2)

t2 = (19491241841, 2)

id(t1[0]) == id(t2[0])
```

```
False
```

```
t1 = (257, 2)

t2 = (257, 2)

id(t1[0]) == id(t2[0])
```

```
False
```

```
t1 = (256, 2)

t2 = (256, 2)

id(t1[0]) == id(t2[0])
```

```
True
```

```
list_3d = [[[i+j+k for k in range(3)] for j in range(3)] for i in range(3)]
```

```
print(list_3d)
```

```
[[[0, 1, 2], [1, 2, 3], [2, 3, 4]], [[1, 2, 3], [2, 3, 4], [3, 4, 5]], [[2, 3, 4], [3, 4, 5], [4, 5, 6]]]
```

```
fruits = ["apples", "bananas", "strawberries"]

res = [fruit.upper() for fruit in fruits]
```

```
print(fruits)
```

```
['apples', 'bananas', 'strawberries']
```

```
print(res)
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```

```
fruits = res
```

```
print(fruits)
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```