Lecture09

February 20, 2024

[1]:
```
# lambda, functions, some important function
# map, filter, zip, sorted, min, max
```

[11]:
```
#,classes, attributes, methods,
# important methods, __init__, __str__, __doc__
# statics, inheritance
```

0.1 Control Statements

[15]:
```
# if, if else, if elif else, nested if
# while, for, break, continue
```

[16]:
```
x=10
if x==10:
    print('this is ten')
elif x<10:
    print('this is less than ten')
else:
    print('This is greater than ten')
```

```
this is ten
```

0.1.1 for loop

[17]:
```
# loop through characters
for c in'welcome':
    print(c)
```

```
w
e
l
c
o
m
e
```

[19]:
```
L=['hasan', True, 2000, 3000, False, 3.2,' alma', [1,2,3]]
for x in L:
    print(x)
```

```
hasan
True
2000
3000
False
3.2
 alma
[1, 2, 3]
```

[22]:
```
# loop through range
# range(start, end, step)
for i in range(5, 20, 2):
    print(i)
```

```
5
7
9
11
13
15
17
19
```

[24]:
```
for i in range(len(L)):
    print(i, L[i])
```

```
0 hasan
1 True
2 2000
3 3000
4 False
5 3.2
6  alma
7 [1, 2, 3]
```

[26]:
```
# loop through keys of a dictionary
d={'hasan': 2000, 'alma':2500, 'mike':4000, 'tara':5000}
for k in d.keys():
    print(k)
```

```
hasan
alma
mike
tara
```

[27]:
```
# loop through values of a dictionary
d={'hasan': 2000, 'alma':2500, 'mike':4000, 'tara':5000}
for k in d.values():
    print(k)
```

```
2000
2500
4000
5000
```

[28]:
```
# loop through items of a dictionary
d={'hasan': 2000, 'alma':2500, 'mike':4000, 'tara':5000}
for k in d.items():
    print(k)
```

```
('hasan', 2000)
('alma', 2500)
('mike', 4000)
('tara', 5000)
```

[31]:
```
# loop through items of a dictionary
d={'hasan': 2000, 'alma':2500, 'mike':4000, 'tara':5000}
for k,v in d.items():
    print(k, '-' , v)
```

```
hasan - 2000
alma - 2500
mike - 4000
tara - 5000
```

0.1.2 list comprehension

example-1

[32]:
```
s=[]
for e in employees:
    s.append(e[1])
s
```

[32]: [20, 23, 40]

[36]:
```
[e[1] for e in employees]
```

[36]: [20, 23, 40]

[38]:
```
list(map(lambda e:e[1], employees))
```

[38]: [20, 23, 40]

[33]:
```
names
```

[33]:
```
['hasan', 'sara', 'alma']
```

[34]:
```
new_names=[]
for n in names:
    new_names.append(n.upper())
new_names
```

[34]:
```
['HASAN', 'SARA', 'ALMA']
```

[35]:
```
[n.upper() for n in names]
```

[35]:
```
['HASAN', 'SARA', 'ALMA']
```

[39]:
```
list(map(lambda n:n.upper(), names))
```

[39]:
```
['HASAN', 'SARA', 'ALMA']
```

example-2

[40]:
```
employees
```

[40]:
```
[('james', 20, 2000), ('sara', 23, 2500), ('alma', 40, 1500)]
```

[41]:
```
# filter-like
[e for e in employees if e[1]>20]
```

[41]:
```
[('sara', 23, 2500), ('alma', 40, 1500)]
```

[43]:
```
# lambda(filter))-like
[e[0] for e in employees if e[1]>20]
```

[43]:
```
['sara', 'alma']
```

[44]:
```
[n if n[0]=='h' else n.upper() for n in names]
```

[44]:
```
['hasan', 'SARA', 'ALMA']
```

[45]:
```
# [n for n in names]
# [n.upper() for n in names]
# [n for n in names if n[0]=='h']
# [n if n[0]=='h' else n.upper() for n in names]
```

0.2 generators

[53]:
```
numbers=[1,2,3,4,5]
```

[4]:
```
def f1(list_of_numbers):
    L=[]
    for n in list_of_numbers:
        L.append(n*n)
    return L
```

[59]:
```
f1(numbers)
```

[59]:
```
[1, 4, 9, 16, 25]
```

[60]:
```
def f2(list_of_numbers):
    return [n*n for n in list_of_numbers]
```

[61]:
```
f2(numbers)
```

[61]:
```
[1, 4, 9, 16, 25]
```

[62]:
```
f3=lambda list_of_numbers:  [n*n for n in list_of_numbers]
```

[63]:
```
f3(numbers)
```

[63]:
```
[1, 4, 9, 16, 25]
```

[2]:
```
L=list(range(1000_000))
```

[20]:
```
%%timeit
f1_result=f1(L)
```

```
47.6 ms ± 3.47 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[8]:
```
def g1(list_of_numbers):
    for n in list_of_numbers:
        yield n*n
```

[21]:
```
%%timeit
g1_result=g1(L)
```

```
190 ns ± 6.97 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

[18]:
```
next(g1_result)
```

[18]:
```
64
```

[26]:
```
# [v*v for v in L]  # return list
[v*v for v in L].__sizeof__()
```

[26]: 8448712

[27]:
```
# (v*v for v in L)    # returns generator
(v*v for v in L).__sizeof__()
```

[27]: 192

[28]:
```
map(lambda v:v*v, L)
```

[28]: <map at 0x7f8a386a78e0>

0.3 Exceptions

[37]:
```
# L*L
# L[200000000]ong')
# print('this will never printed')
```

[41]:
```
raise TypeError('waeit .. thios is wrong')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 1
----> 1 raise TypeError('waeit .. thios is wrong')

TypeError: waeit .. thios is wrong
```

[43]:
```
names=['james','sarah','mike','hasan','tara','william']
```

[45]:
```
for n in names:
    if n=='hasan':
        raise Exception('hasan is not welcomed')
    print('hi..', n)
```

```
hi.. james
hi.. sarah
hi.. mike
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[45], line 3
      1 for n in names:
      2     if n=='hasan':
----> 3         raise Exception('hasan is not welcomed')
```

```
      4     print('hi..', n)

Exception: hasan is not welcomed
```

[59]:
```
# try except
def div(x,y):
    try:
        result=x/y
    except ZeroDivisionError:
        result='not possible .. we can not divide by zero'
    except TypeError:
        result = int(x) / int(y)
    else:
        print('well done.. no problems')
    finally:
        print('thank you...')
    return result
```

[60]:
```
div(4,'2')
```

```
thank you…
```

[60]: 2.0

[61]:
```
div(4,2)
```

```
well done.. no problems
thank you…
```

[61]: 2.0

[62]:
```
# try:
#     write to main_file
# except File_Not_Found:
#     write_to_back_file
# else:
#     report this to the manager
# finally:
#     add an event to the log file
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: