# Lecture08

February 20, 2024

[1]:
```
# Data structure
# list, tuple, sets, dict
```

## 0.1 functions in python

[ ]:
```

```

[18]:
```
def myfun(x,y=0):
    return x+y, x-y, x*y
```

[28]:
```
result=myfun(10,5)
type(result)
```

[28]:
```
tuple
```

[20]:
```
myfun(10)
```

[20]:
```
(10, 10, 0)
```

[22]:
```
a,s,m=myfun(10,5)
```

[23]:
```
a
```

[23]:
```
15
```

## 0.2 some important functions

[2]:
```
# len, min, max, sum, sorted
```

[4]:
```
names=['hasan', 'sara', 'robert', 'william', 'anas', 'sam']
sorted(names)
```

[4]:
```
['anas', 'hasan', 'robert', 'sam', 'sara', 'william']
```

[9]:
```
sorted(names, key=len)
```

[9]:
```
['sam', 'sara', 'anas', 'hasan', 'robert', 'william']
```

```
[10]: def get_last_char(n):
          return n[-1]
```

```
[12]: sorted(names, key=get_last_char)
```

```
[12]: ['sara', 'william', 'sam', 'hasan', 'anas', 'robert']
```

```
[34]: def get_last_char(n):
          return n[-1]
```

```
[35]: get_last=lambda n:n[-1]
      get_last('james')
```

```
[35]: 's'
```

```
[36]: def myfun(x,y):
          return x+y, x-y, x*y
```

```
[37]: my_lambda=lambda x,y: (x+y, x-y,x*y)
      my_lambda(10,5)
```

```
[37]: (15, 5, 50)
```

```
[ ]:
```

#### 0.2.1 map

```
[29]: names
```

```
[29]: ['hasan', 'sara', 'robert', 'william', 'anas', 'sam']
```

```
[32]: list(map( len, names))
```

```
[32]: [5, 4, 6, 7, 4, 3]
```

```
[33]: list(map(get_last_char, names))
```

```
[33]: ['n', 'a', 't', 'm', 's', 'm']
```

```
[38]: list(map(lambda n:n[-1],names))
```

```
[38]: ['n', 'a', 't', 'm', 's', 'm']
```

**Example-1**

```
[39]: employees = [('hasan', 40, 5000), ('sara', 20, 6000), ('alma', 25, 7000)]
```

```
[42]: # write one line code to sort this employees by age
      sorted(employees, key=lambda e:e[1])
```

```
[42]: [('sara', 20, 6000), ('alma', 25, 7000), ('hasan', 40, 5000)]
```

```
[45]: # write one line code that will find the total sum of all employees salaries
      sum(map(lambda e:e[-1], employees))
```

```
[45]: 18000
```

### 0.2.2 filter

```
[46]: names
```

```
[46]: ['hasan', 'sara', 'robert', 'william', 'anas', 'sam']
```

```
[47]: list(filter(lambda n:len(n)>3 ,names))
```

```
[47]: ['hasan', 'sara', 'robert', 'william', 'anas']
```

```
[48]: # filter all employees based on age (age <30)
      employees
```

```
[48]: [('hasan', 40, 5000), ('sara', 20, 6000), ('alma', 25, 7000)]
```

```
[49]: list(filter(lambda e:e[1]<30,employees))
```

```
[49]: [('sara', 20, 6000), ('alma', 25, 7000)]
```

#### Example-2

```
[50]: # write one line code to find names of employees with salary less than 7000
      list(map(lambda x:x[0], filter(lambda e:e[-1]<7000,employees)))
```

```
[50]: ['hasan', 'sara']
```

### 0.2.3 Iterators

```
[54]: names
```

```
[54]: ['hasan', 'sara', 'robert', 'william', 'anas', 'sam']
```

```
[62]: i=iter(names)
```

```
[53]: type(i)
```

```
[53]: list_iterator
```

[64]:
```
next(i)
```

[64]:
```
'sara'
```

[65]:
```
list(i)
```

[65]:
```
['robert', 'william', 'anas', 'sam']
```

## 1 classes

[106]:
```
class person:
    counter=0
    def say_bye():
        print('bye .....')
    def __init__(self, n='', a=0):
        print('a new object was created')
        self.name=n
        self.age=a
    def say_hi(self):
        print('hi...')
    def __str__(self):
        return 'I am peron, my name is: ' + self.name +' anbd my age is: '␣
 ↪+str(self.age)
```

[107]:
```
p1=person('hasan', 40)
```

```
a new object was created
```

[108]:
```
str(p1)
```

[108]:
```
'I am peron, my name is: hasan anbd my age is: 40'
```

[109]:
```
print(p1)
```

```
I am peron, my name is: hasan anbd my age is: 40
```

[110]:
```
person.say_bye()
```

```
bye …
```

[111]:
```
p2=person()
```

```
a new object was created
```

[112]:
```
p1.name
```

[112]:
```
'hasan'
```

```
[113]: p1.say_hi()
```

```
hi…
```

```
[114]: print(p1)
```

```
I am peron, my name is: hasan anbd my age is: 40
```

```
[115]: p1=person()
        p2=person()
        p3=person()
```

```
a new object was created
a new object was created
a new object was created
```

```
[117]: L=[p1,p2,p3]
```

```
[121]: class employee(person):
            def __init__(self, n='',a=0, s=100):
                super().__init__(n,a)
                self.salary=s

            def add(self, x,y):
                return x+y
```

```
[122]: e1=employee()
        e1.add(10,5)
```

```
a new object was created
```

```
[122]: 15
```

```
[124]: e1.salary
```

```
[124]: 100
```

### 1.0.1 Example-3

```
[129]: s1='welcome'
        s1=str('welcome')
```

```
[130]: class mystr(str):
            def remove_first_last(self):
                return self[1:-1]
```

```
[131]: s2=mystr('welcome')
```

```
[132]: s2.remove_first_last()
```

```
[132]: 'elcom'
```

```
[133]: s1.remove_first_last()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[133], line 1
----> 1 s1.remove_first_last()

AttributeError: 'str' object has no attribute 'remove_first_last'
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```