Oblig 3

1: Dictionaries er en måte å behandle data på i python. I dict lagres dataen i «par», som i dette eksempelet:

```
student = {
    "first name" : "Ola",
    "last name" : "Nordmann",
    "favourite course" : "Programmering 1"
}
```

Her ser man at dataen blir holdt i et «key:value» par, som «first name» : «Ola», hvis man da ville printet «Ola» må man da skrive:

```
print(student["first name"])
```

fordi value «ola» er bundet til key «first name» i dictionary «student»

forskjellen mellom lister og dict i python er nettopp dette key:value forholdet, hvor i list er det bare verdier, og man skiller mellom disse på den numeriske posisjonen gitt til hver enkelte verdi (0, 1, 2 osv) eksempel:

```
student = ["Ola","Nordmann","Programmering 1"]
```

her er en liste som inneholder samme data som dict, hvis jeg skulle ha printet «Ola», måtte jeg ha gjort det sånn her:

```
print(student[0])
```

fordi «0» er posisjonen til «Ola» i listen

2:

Funksjoner er i essens kodesnutter som bare kjøres når de blir kalt på. Som for eksempel:

```
def print_hello():
    print("hello")
```

vil bare kjøres hvis:

```
print_hello()
```

er et annet sted i koden. Python har mange innebygde funksjoner, som for eksempel «print()». Egendefinerte funksjoner må defineres i koden, for å så bli kalt inn senere i koden.

Funksjoner kan også ta data som input-parametre og gjøre noe med det, som for eksempel:

```
def print_hello(x):
    print(x)
```

vil da printe x-en, så:

```
print_hello("hallo")
```

vil da føre til at funksjonen printer «hallo», fordi det er input-parameteret.

Funksjoner trenger ikke å ha noen umiddelbart synlige resultater, som for eksempel kan funksjoner brukes til å manipulere lister eller dict. Som for eksempel:

```
list = ["a","b"]
print(list)
def edit_list():
    list[0] = "c"
edit_list()
print(list)
```

vil først printe ['a', 'b'], fordi det er det listen originalt er.

Etter funksjonen har blitt kalt, vil den printe : ['c', 'b']

Funksjoner gjør at programmereren kan gjøre «gjøre»-delen av koden sin(det vil si den biten av kode som faktisk står bak for at ting skjer, som for eksempel en main loop ellerno) blir mye ryddigere, og det forhindrer at python må lese mellom masse linjer om og om igjen i en løkke, hvis istedenfor hele kodeblokken, det bare står en funksjon-call.