## Deel 1 – maak functies

Bekijk eerst de video's functies 1 t.e.m. functies 7 op www.youtube.be/dve845

Voor deze opgave zijn er twee bestanden. Het bestand **opgave2.py** en het bestand **opgave2tester.py**. Maak een nieuw project en zet beide bestanden in de projectfolder. Hernoem het bestand **opgave2.py** naar **opgave2opl.py**.

Het laatste bestand bevat de reeds gedefinieerde automatische testen voor de functies die moeten geschreven worden in het eerste bestand. Het is de bedoeling dat alle testen slagen. Probeer niet alle testen in één keer aan te pakken. Werk ze test per test (en dus functie per functie) af.

Zelf testen toevoegen mag, maar de bestaande laat je ongemoeid. Bekijk alle testen eens in vogelvlucht om een idee te hebben van welke functionaliteit gevraagd wordt, want het is best mogelijk dat sommige functies nuttig kunnen zijn voor het definiëren van andere. De volgorde van de testen volgt dus niet noodzakelijk de beste volgorde voor implementatie.

Voor deze opgave mag gebruik gemaakt worden van volgende zaken

- de functie **`len()`**
  *len(x)* geeft de lengte van de string of de array x terug
- de functie **`ord()`**
  *ord(x)* geeft de ascii-waarde van het karakter x terug
  hoewel deze functie nodig is, laat Python wel relationele operaties < en > toe op karakters
- de functie **`chr()`**
  *chr(x)* geeft het ascii-karakter terug dat hoort bij de ascii-waarde x
- de functie **`append()`**
  *x.append(y)* de waarde y wordt toegevoegd aan de tabel x, waardoor x groter wordt
  let op de puntnotatie, het wordt volgend semester duidelijk waar dit vandaan komt
- de **`[]`** operator op strings
  *x[i]* geeft het i-de karakter terug van de string x
- de functie **`isWhitespace()`**
  *isWhitespace(x)* geeft True terug als x een onzichtbaar karakter is, anders False
  deze functie is gegeven in opgave2.py en kun je gewoon gebruiken as-is
  (we noemen een functie die True/False terugkeert een logische functie).

Gebruik van andere functies is niet nodig maar bovendien 'verboten'! Alles wat je nodig hebt maak je in deze oefeningen zelf. Ter herinnering, het doel hier is het oefenen van functies, niet het aanleren van Python.

## Deel 2 – gebruik de functies (met dank aan Bernard Samyn)

Onderstaande opgaves maken voor hun oplossing gebruik van de functies die je in deel 1 ontwikkelde. Voor dit tweede deel zijn er geen testen. Maar het staat je vrij er zelf te maken voor eventuele functies die je wil definiëren voor deze opgaves.

1. Schrijf een programma, dat het aantal letters en cijfers bepaalt van een ingelezen zin.

***Voorbeeld**:*

```
Geef een zin in: Op woensdag 16/11/2016 vindt het 2de werkcollege plaats.
Aantal cijfers: 9
Aantal letters: 37
```

2. Schrijf een programma dat het saldo bepaalt op een bankrekening na een reeks transacties, die op afzonderlijke regels via console input worden ingegeven en afgesloten worden met een lege regel. Het initiële saldo is 0. De transacties worden ingegeven onder volgende vorm:
**`D 300`** (d.w.z. Deposit 300 – Stort 300)
**`W 100`** (d.w.z. Withdraw 100 – Haal 100 af)
Foutieve transacties, zoals **`D200`** [geen spatie] of **`X 50`** [geen D of W] of **D 100 205** [meer dan één bedrag] (*), evenals negatieve bedragen, zoals **`d -100`** of **`W -50`** (§), worden genegeerd.
Bovendien mag het saldo nooit negatief worden: indien een operatie het saldo negatief zou maken, dan wordt die operatie eveneens genegeerd (#).

***Voorbeeld**:*
Volgende input

```
D 400
d100          (*)
d 100
w 750         (#)
W 150
D 100 150     (*)
D 50
W -200        (§)
W 100
d -50         (§)
D 200
F 50          (*)
W 300
```

levert als resultaat:

```
200
```

---

3. Een programma vraagt een cijfer *d* en een aantal *n* en drukt vervolgens volgende som af op het beeldscherm: *d + dd + ddd + … + dddddd* (het aantal termen is *n*).

***Voorbeeld**:* als d = 9 en n = 4, dan is de uitkomst 11106 (immers: 9 + 99 + 999 + 9999 = 11106)

```
Cijfer: 9
Aantal: 4
11106
```

---

4. Een robot beweegt in een vlak vanaf het startpunt (0, 0). De robot kan in vier richtingen bewegen: Up (U), Down (D), Left (L) en Right (R) met een aantal stappen. Schrijf een programma, dat de afstand van de robot tot het beginpunt berekent, nadat een aantal bewegingen zijn ingegeven (sluit af met een lege regel). De afstand wordt afgerond naar het dichtstbijzijnde gehele getal.

***Voorbeeld**:*
Volgende input

```
U 1
D 3
u 2
L 3
r 2
U 2
```

levert als resultaat

```
2
```

---

5. Een reeks paswoorden, gescheiden door een komma, moet gecontroleerd worden op geldigheid. Van elk paswoord in de reeks wordt vermeld of het ‘correct’ of ‘foutief’ is. Een paswoord is correct als het

- minstens 1 hoofdletter bevat
- minstens 1 kleine letter bevat
- minstens 1 cijferteken bevat
- minstens 1 teken bevat uit volgende reeks van drie tekens: $#@
- minimaal 6 tekens bevat
- maximaal 12 tekens bevat
- geen spatie(s) bevat

***Voorbeeld**:*
De inputstring

```
ABd1234@1,a F1#,12345678,aB$1234,9z5D@,1a2B3#,Abcd1234#xyz,a$1d6Tt33xsq#9
```

levert als resultaat:

```
ABd1234@1: correct
a F1#: foutief
12345678: foutief
aB$1234: correct
9z5D@: foutief
1a2B3#: correct
Abcd1234#xyz: correct
a$1d6Tt33xsq#9: foutief
```