Beginselen van Programmeren: Oefenzitting 4-5

functies – bestanden – collecties (vervolg) – probleem oplossen – commentaar

Ontwerptip: Vermijd functies die zelf om invoer van de gebruiker vragen. In dergelijke gevallen is het beter om deze invoer via parameters door te geven, en de code voor interactie met de gebruiker in een `main()` *functie onder te brengen. Hetzelfde geldt voor afdrukstatements op de standaarduitvoer. Vaak is het beter om het resultaat via een* `return` *statement terug te geven aan de aanroeper van de functie.*

1. Duid voor elke variabele in de programmacode `oefz4-scope.py` aan wat zijn scope is.
2. Schrijf een functie die string met een piramide opbouwt. De functie verwacht twee argumenten of parameters: de hoogte van de piramide en het af te drukken symbool. De tweede parameter is optioneel en indien niet opgegeven dient het symbool `'#'` gebruikt te worden. Schrijf een main programma dat deze functie gebruikt.
3. Schrijf een programma dat een tekstbestand inleest. Het programma schrijft de inhoud van het bestand lijn per lijn weg naar een nieuw bestand, maar voegt lijnnummers toe.
   Indien het invoerbestand volgende tekst bevat:
   ```
   Mary had a little lamb
   whose fleece was white as snow.
   And everywhere that Mary went,
   the lamb was sure to go!
   ```
   dan bevat het uitvoerbestand deze tekst:
   ```
   /* 1 */ Mary had a little lamb
   /* 2 */ whose fleece was white as snow.
   /* 3 */ And everywhere that Mary went,
   /* 4 */ the lamb was sure to go!
   ```
   Vraag de gebruiker om de naam van beide bestanden.
4. Schrijf een programma dat 20 dobbelsteenworpen uitvoert en het resultaat daarvan in een lijst bewaart. Druk die lijst af en markeer de langste reeks opeenvolgende gelijke worpen. Indien er meerdere langste reeksen zijn, markeer dan enkel de eerste.
   Voorbeeld: `1 2 5 5 3 1 2 4 3 (2 2 2 2) 3 6 5 5 6 3 1`
   (*Tip*: Denk eerst na over hoe je de verschillende taken in functies zal opdelen!)
5. (**UOVT**) Schrijf een set functies die voor lijsten van gehele getallen de volgende taken uitvoeren. Zorg ervoor dat je functies de parameterwaarden niet wijzigen, maar dat je de gevraagde lijst als resultaat teruggeeft.
   a. `swap(lijst)`: Verwissel het eerste en laatste element van de lijst.
   b. `shift(lijst)`: Schuif alle getallen 1 plaats naar rechts en verplaats het laatste getal naar het begin van de lijst. *Voorbeeld*: de lijst [1, 4, 9, 16, 25] wordt dus [25, 1, 4, 9, 16].
   c. `replaceEvenWithZeros(lijst)`: Vervang alle even getallen in een lijst door het getal 0.
   d. `moveEvenToFront(lijst)`: Verplaats alle even getallen naar het begin van de lijst, zorg ervoor dat de onderlinge volgorde van de even (en de oneven) getallen niet wijzigt.
   e. `reverse(lijst)`: Geef een lijst terug met de getallen in omgekeerde volgorde.

6. Schrijf functies die toelaten om een woordenschat op te bouwen op basis van de regels die in een tekstbestand staan:
   a. Schrijf een functie `getWoorden(regel)` die voor een gegeven regel de set met woorden die erin voorkomen teruggeeft. Gebruik de voorgedefinieerde functie `split()` die een string opsplitst in een sequentie van woorden. Implementeer ook de functie die woorden ontdoet van leestekens en hoofdletters:
   ```
   import string
   def cleanWord(w):
     return w.lower().strip(string.punctuation)
   ```
   b. Schrijf een functie `leesBestand(naam)` die een gegeven tekstbestand inleest en een lijst van regels teruggeeft.

   Schrijf een programma dat je functies gebruikt om de woordenschat op te bouwen van een tekstbestand. Na het verwerken van een regel drukt je programma de huidige grootte van de woordenschat af, alsook welke woorden er werden toegevoegd. Wanneer de laatste zin werd verwerkt drukt het programma vervolgens de woorden af die in meer dan 50% van de zinnen voorkomen.
   Voorbeelduitvoer:
   ```
   Verwerkte zin: 'Java is crazy.'
   Grootte van de nieuwe woordenschat: 3
   Nieuwe woorden toegevoegd: {'crazy', 'java', 'is'}
   Verwerkte zin: 'Python is really crazy!'
   Grootte van de nieuwe woordenschat: 5
   Nieuwe woorden toegevoegd: {'really', 'python'}
   KLAAR
   Woorden die in meer dan 50% van de zinnen voorkomen:
   crazy is
   ```
7. Schrijf een functie die controleert of een 4 *x* 4 matrix een magisch vierkant is. Een magisch vierkant is een *n x n* matrix gevuld met de getallen 1, 2, 3, ... , n^2 waarbij de som van de getallen op elke rij, elke kolom en de twee diagonalen telkens dezelfde waarde heeft. Een matrix in Python is een lijst van lijsten.
   Schrijf een programma dat 16 waarden in een matrix inleest en test of de matrix een magisch vierkant is. Je functie controleert dus twee zaken:
   a. of alle getallen 1, 2, 3, ... , 16 ingevoerd zijn,
   b. en of de som van alle rijen, kolommen en diagonalen dezelfde waarde heeft.
8. Schrijf een functie voor het coderen en decoderen van boodschappen via het Vigenère codeersysteem. In dit systeem wordt een codewoord gebruikt om tekst te (de)coderen i.p.v. een vast cijfer zoals bij de Caesar codering. Het codewoord (bvb. `'LEMON'`) wordt zoveel herhaald als nodig en de encryptie wordt als volgt bepaald:
   ```
   Attack at dawn!
   +++++++++++++++
   LEMONLEMONLEMON
   ===============
   Lxfopv mh oeib!
   ```
   Waarbij voor de letters `A`, `B`, `C`, ... de waarden 0, 1, 2, ... gebruikt worden.[1] Ook voor de letters `a`, `b`, `c`, ... worden dezelfde waarden gebruikt. Het maakt dus niet uit of je codewoord "`LEMON`" of "`LeMoN`" is. Zorg er wel voor dat hoofd- en kleine letters in het invoerbestand ook overeenstemmen met hoofd- en kleine letters in het uitvoer-bestand.
   Schrijf één functie die zowel de codering als decodering kan uitvoeren, voorzie dus een parameter die de modus bepaalt.
   <u>Uitbreiding:</u> Schrijf functies voor het (de)coderen van een tekstbestand.

[1] Ter info: In UNICODE worden de letters *A-Z* voorgesteld door de waarden 65-90, de letters *a-z* door de waarden 97-122. In principe heb je deze informatie niet nodig.