Programmeren in Python

Deel 5: functies - Deel 6: recursie

S. Vanspeybroeck

Sint-Pieterscollege Jette

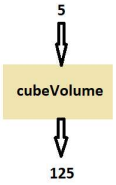
Schooljaar: 2021-2022

What to do?

Voorbeelden

```
# functie definiëren
def cubeVolume(sideLength):
    volume = sidelength ** 3
    return volume

# functie uitvoeren
cubeVolume(5)
```



syntax

def *functieNaam*(*param_1*,..., *param_n*)**:**
    statements
    return *waarde*

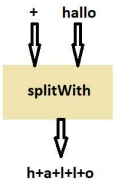
Voorbeelden

```
# functie definiëren
def sayHello():
    print("Hello!")

# functie uitvoeren
sayHello()
```

```
# functie definiëren
def splitWith(symb, word):
    split = ""
    for letter in list(word):
        split = split + letter + symb
    return split[0: len(split) - 1]

# functie uitvoeren
splitWith("+", "Hallo")
```



Oefeningen

*

1. Beschouw de funties:

```
import math

def f(x):
    return g(x) + math.sqrt(h(x))

def g(x):
    return 4 * h(x)

def h(x):
    return x*x + k(x) - 1

def k(x):
    return 2 * (x + 1)
```

Bepaal: a) f(2) b) g(h(2)) c) k(g(2)+h(2))

Oefeningen

*

2. Bepaal: **splitWith**("+", **splitWith**("+", "Hola"))
3. (Vermoeden van Collatz) Beschouw de functie:

```
def f(n):
    #herhaal zolang n > 1:
        # Deel door 2 als n even is:
        # Vermenigvuldig n met 3 en tel 1 bij
op als n oneven is
    return n
```

- Vervolledig de functie met een loop.
- Bepaal voor f(1), f(2) en f(10) de waarden van n bij elke lus.

Opmerking: de functie f(n) start een lus die steeds n deelt door 2 als n even is en vemenigvuldigt met 3 (plus 1) als n oneven is TOTDAT n <= 1. Het *vermoeden van Collatz* is een **onbewezen bewering** dat zegt dat de lus voor elke natuurlijk getal n zal stoppen. M.a.w f(n) zal geen oneindige lus uitvoeren.

Oefeningen

**

4. Schrijf een functie

   def **abs**(getal):

   die de absolute waarde van een getal teruggeeft. Bv. abs(-5) geeft 5 en abs(5) geeft 5.
5. Voor een cilinder met straal r en hoogte h, schrijf de functie
   a) **cilinderVol**(r, h),
   b) **cilinderOpp**(r, h)

   die : [a)] het volume, [b)] de oppervlakte van de cilinder geeft. Gebruik de module **math** om de variabele **math.pi** (π) te kunnen gebruiken.
6. Schrijf een functie

   def **maximum**(a,b,c):

   die het maximum van de getallen a,b en c bepaalt. Test je functie uit.

Oefeningen

** _ ***

7. Schrijf een functie

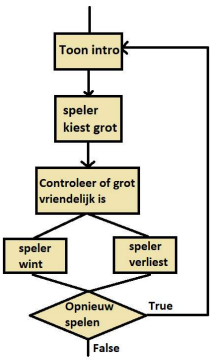
   def **telWoorden**(string):

   dat het aantal woorden in een string teruggeeft. Woorden zijn in deze oefeningen gescheiden door een spatie. Bv.

   telWoorden("Marie heeft een goudvis.")

   geeft 4 terug.

Dragon Realm: sample run




```
Je bevindt je in een land vol draken. Voor jou zie je twee grotten:
 In de ene grot, is een vriendelijke draak die zijn schat met jou wil delen.
 In de andere grot is een hebzuchtige en hongerige draak die je zal opeten.
Welke grot ga je binnen? (1 of 2): 2
Je nadert de grot...
Het is hier donker en akelig...
Een gigantische draak verschijnt voor jou! Hij opent zijn kaken en...
Geeft jou haar schat!
Wil je opnieuw spelen? (ja of nee) nee
```

Dragon Realm: overzichtelijk werken

```
import random
import time

spelOpnieuw = "ja"
while spelOpnieuw == "ja" or spelOpnieuw ==  "j":
    # Toon intro
    print("Je bevindt je in een land vol draken. Voor jou zie je twee grotten:\n
      In de ene grot, " +
          "is een vriendelijke draak die zijn schat met jou wil delen.\n In de
     andere grot is een " +
          "hebzuchtige en hongerige draak die je zal opeten.")
    #kies een grot:
    grot = ""
    while grot != 1 and grot != 2:
        grot = int(input("Welke grot ga je binnen? (1 of 2): "))
    #controleer grot
    print("Je nadert de grot...")
    time.sleep(2)
    print("Het is hier donker en akelig...")
    time.sleep(2)
    print("Een gigantische draak verschijnt voor jou! Hij opent zijn kaken en...
     ")
    time.sleep(2)

    vriendelijkeGrot = random.randrange(1, 3)

    if vriendelijkeGrot == grot:
        print("Geeft jou haar schat!")
    else:
        print("Eet je op in één hap!")
    #Vraag om opnieuw te spelen
    spelOpnieuw = input("Wil je opnieuw spelen? (ja of nee) ")
```


Dragon Realm: main()

```
import random
import time

##definities van de functies:
...

def main():
    spelOpnieuw = "ja"
    while spelOpnieuw == "ja" or spelOpnieuw ==  "j":
        toonIntro()
        grot = kiesGrot()
        controleGrot(grot)

        spelOpnieuw = input("Wil je opnieuw spelen? ")

# Voer het spel uit
main()
```


Dragon Realm: toonIntro() - kiesGrot()

```
##definities van de functies:
def toonIntro():
    print("Je bevindt je in een land vol draken. Voor
    jou zie je twee grotten:\n In de ene grot, " +
          "is een vriendelijke draak die zijn schat
    met jou wil delen.\n In de andere grot is een " +
          "hebzuchtige en hongerige draak die je zal
    opeten.")

#Vraag de speler naar een getal 1 of 2 (grot = 1 of
    grot = 2) TOT ZOLANG de gebruiker geen van beide
    getallen ingeeft.
def kiesGrot():
    grot = ""
    ...
```

controle

Vervolledig de functie kiesGrot().

Dragon Realm: controleGrot()

```
...

def controleGrot(grot):
    print("Je nadert de grot...")
    time.sleep(2)
    print("Het is hier donker en akelig...")
    time.sleep(2)
    print("Een gigantische draak verschijnt voor jou!
    Hij opent zijn kaken en...")
    time.sleep(2)

    vriendelijkeGrot = random.randrange(1, 3)

    if vriendelijkeGrot == grot:
        print("Geeft jou haar schat!")
    else:
        print("Eet je op in één hap!")

def main():
...
```


Oefeningen

**

1. Schrijf een programma die de gebruiker vraagt naar een *string* zin en het aantal klinkers, medeklinkers en overige tekens (inclusief spaties) print. Werk als volgt:

a) Schrijf een functie **aantalKlinkers(zin)** die het aantal klinkers in de zin geeft. Tip:

```
def aantalKlinkers(zin):
    # variabele die het aantal klinkers bijhoudt:
    KLINKERS = ["a", "e", "i", "o", "u"]
    aantal = 0
    ##Pas aan:
    # voor elke letter in zin:
        if letter.lower() in KLINKERS:
            ...
    return ...
```

Doe hetzelfde voor de functie **aantalMedeKl(zin).**

Oefeningen

**

b) Schrijf een functie **overigeSymbolen(zin)** die het aantal overige tekens (geen letters) teruggeeft in een zin.

b) Schrijf de **main()** functie die de gebruiker vraagt naar de een zin. En de voorgaande functies gebruikt om het aantal klinkers, medeklinkers en overige tekens af te printen. TEST JE CODE.

Oefeningen

***

2. Schrijf een programma die de gebruiker vraagt naar een *integer* n en de eerste n priemgetallen geeft. Werk als volgt:

a) Schrijf een functie **aantalPriem()** die de gebruiker vraagt naar het aantal gewenste priemgetallen (**n**) en dit getal teruggeeft.

b) Schrijf een functie **isPriem(x)** die controleert of een getal, **x** priem is. Indien zo geeft het **True** terug anders **False.**

c) Schrijf de **main()** functie die de voorgaande functies gebruikt om de eerste **n** priemgetallen af te printen.

Recursie: voorbeelden

- Voorbeeld 1:

```
def printDriehoek(hoogte):
    if hoogte < 1: return
    else:
        print("[]" * hoogte)
        printDriehoek(hoogte - 1)
```

- Voorbeeld 2:

```
def printDriehoek(hoogte):
    if hoogte < 1: return
    else:
        printDriehoek(hoogte - 1)
        print("[]" * hoogte)
```


Recursie: voorbeelden

controle

1. Wat is het verschil tussen voorbeeld 1 en 2?
2. Wat gebeurt er als de if - statement wordt weggelaten?
3. Beschouw de recursieve functie:

```
def mystery(n):
    if n<= 0: return 0
    return n + mystery(n - 1)
```

Bepaal mystery(4).

Recursie: voorbeelden

- Voorbeeld 3: (Fibonacci)

Rij: $1, 1, 2, 3, 5, 8, 13, 21...$

Voorschrift: $u_{n+2} = u_{n+1} + u_n \quad \forall n > 0 \text{ en } u_1 = u_2 = 1$

```
def fib(n):
    ...
        return ...
    ...
        return ...
```

controle

4. Schrijf een programma die naar een waarde *n* vraagt en de eerste *n* fibonacci getallen afprint.

Recursie: voorbeelden

- Voorbeeld 4: (Permutaties)

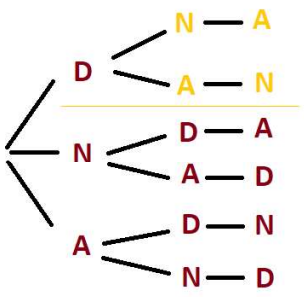


Figure: permutaties van DNA

RECURSIE:

	D	+
	N	+
	A	+
Alg.:	i^{de} letter	+

Recursie: permuties -code

```
##Bepaal alle permutaties van woord:
# return: een lijst, result met alle permutaties
def permutaties(woord):
    result = []
    #een letter heeft één permutatie: zichzelf
    if len(woord) == 1:
        result.append(woord)
        return result
    else:
    # doorloop de letters van het woord
    for i in range(len(woord)):
        # overige letters
        overig = woord[:i] + woord[i+1:]
        #permutaties van overige letters
        overigPermutaties = ...
        #plaats verw. letter vooraan elke perm. v...
        ...
        ...
    # geef de permutaties terug:
    return result
```

Recursie: permuties -code

controle

5. Vervolledig de code waar aangegeven.
6. Gebruik de functie permutaties om de string *python* te permuteren.
7. Kan je de code toepassen op een lijst? (Bv. $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$)

Recursie: oefeningen

******

1. Schrijf een recursieve functie

   def **cijfersIN**(n):

   die telt hoeveel cijfers er in het **positief** getal n met datype **integer** voorkomen.

   Bv. cijfersIN(5033) geeft 4.

   Tip: Als $n < 10$ dan bestaat n uit 1 cijfer. Anders, heeft n één cijfer meer dan n//10.
2. Schrijf een recursieve functie

   def **macht**(a, n):

   die de macht a^n berekent. Tip: $a^n = a.a^{\cdots}$

Recursie: oefeningen

******

3. Schrijf een recursieve functie

   def **reverse**(string):

   die de volgorde van de karakters omdraait.

   Bv. reverse(wolf) geeft flow.

   Tip: draai de substring om die begint vanaf het 2de karakter en voeg er achter het eerste karakter toe. Bv. van "w" + "olf" bepaal eerst het omgekeerde van "olf", ("flo") en voeg vervolgens "w" er achteraan toe.
4. Schrijf een functie

   def **isPalindroom**(string):

   die test of een woord een *palindroom* (= gelijk is aan haar omgekeerde).

   Bv. isPalindroom(negen) geeft **True**.

Recursie: oefeningen

***

5. Schrijf een recursieve functie

   def **find**(string, match):

   die controleert of "match" in de string voorkomt.

   Bv. find("mississipi, "sip") geeft True.

   Tip: Als 'string' begint met 'match' ben je klaar. Anders bekijk je de string dat je verkrijgt na het verwijderen van de eerste letter.