Workshop Python voor Wis- en Natuur/Sterrenkundigen

Vrijdag 27 januari 2017, 11:00 - 13:00

In deze workshop maken we kennis met Python door middel van een aantal eenvoudige opgaven. We beginnen met een opdracht om bekend te worden met de mogelijkheden van de interactieve interpreter. Voor de daarop volgende opgaven is het de bedoeling deze uit te werken in een editor en te draaien met behulp van de interpreter. De interactieve interpreter mag hierbij natuurlijk wel als hulpmiddel worden gebruikt!

Een eerste kennismaking

1. Open een terminal-venster en start de Python interpreter op door het commando `python` te geven. Doe vervolgens de volgende oefeningen:

   (a) Maak twee variabelen `a` en `b` aan met een geheel getal of floating-point getal als initiële waarde. Voer vervolgens een aantal berekeningen uit zoals `a + b`, `a * b`, `a ** b` enzovoort. Probeer ook het resultaat op te vangen in een variabele `c`. *Tip: je kunt met "pijl omhoog" terug bij je voorgaande invoeren*.

   (b) Wat gebeurt er als `a` een integer is en `b` een floating-point getal?

   (c) Maak een aantal meer variabelen aan en probeer een paar langere expressies: `(a + b) * c`, `(a ** 3 * c) / d`.

   (d) Ga na dat het optellen van een string en een getal niet werkt, maar het vermenigvuldigen van een string en een getal wel. Kun je het resultaat verklaren?

   (e) **(+)** Probeer ook eens te rekenen met complexe getallen.

   (f) Maak twee variabelen met floating-point getallen en zet deze met behulp van `print` en `.format` netjes op het scherm. Maak het eerste getal 4 vakjes breed met 1 getal achter de komma en het tweede getal 10 vakjes breed met 6 plaatsen achter de komma.

   (g) Gegeven de string `s = "een twee drie vier vijf zes zeven acht"`. Maak met behulp van de *slice* notatie (`s[x:y]`) expressies die de woorden "twee", "vier" en "zeven" isoleren. Voorbeeld: `s[:3]` isoleert het woord "een".

   (h) Gebruik `range` om de volgende getallenreeksen te maken: 0 t/m 9, 1 t/m 10, alle even getallen in 0 t/m 20 en de eerste 10 getallen van de tafel van zes.

   (i) Om de interpreter af te sluiten gebruik je het statement `exit()`.

Zoals je hebt gezien is de interactieve interpreter een handig hulpmiddel. Bij het schrijven van een groot Python programma kun je korte statements en expressies dus in de interactieve interpreter even snel uitproberen om te kijken of deze het beoogde resultaat geven.

Voor de volgende opdrachten is het de bedoeling dat deze worden gemaakt door een programma te schrijven in een editor en het daarna te draaien met de Python interpreter. Maak in je favoriete editor (bijvoorbeeld `gedit` of `kate`) een nieuw bestand: `programma.py` (of welke bestandsnaam je maar wilt). Als je wilt kun je ook kijken of de editor is ingesteld om "net" Python te schrijven: gebruik altijd spaties in plaats van tabs en spring in met 4 spaties. Schrijf vervolgens je programma. Om het programma te draaien geef je het volgende commando in het terminalvenster: `python programma.py`.

2. Schrijf een programma dat de gebruiker om hoogte en breedte vraagt (gebruik hiervoor `int(raw_input())`). Controleer met een `if`-statement of de ingevoerde getallen groter zijn dan nul, zo niet geef een foutmelding. Reken de omtrek en oppervlakte uit en zet de resultaten op het scherm.

3. Definieer de volgende lijst bovenaan je programma:

```
l = [7, 43, 45, 32, 26, 42, 4, 33, 16, 40, 38, 39, 44, 14, 23]
```

   Schrijf een loop om het gemiddelde te berekenen van deze lijst.

4. Schrijf een loop die het gemiddelde berekent van alle elementen in de lijst op een oneven positie. *Hint: gebruik step in* `range`.

5. Vraag de gebruiker om een string in te voeren die bestaat uit een lijst gehele getallen gescheiden door spaties. Vang de invoer op in een string `s`. Je kunt de ingelezen string als volgt omzetten in een lijst van gehele getallen:

```
l = map(int, s.split(" "))
```

   Schrijf een loop die nu alle getallen afdrukt tussen 3 en 44 of deelbaar zijn door 6.

6. **(+)** Tot nu toe moesten we altijd ons programma draaien door Python expliciet aan te roepen: `python programma.py`. We kunnen er ook een echt uitvoerbaar programma van maken, net als C++ programma's. Dit gaat als volgt: in `programma.py`, voeg als de **allereerste** regel in het programma toe:

```
#!/usr/bin/env python
```

   Dit heet de *she-bang regel*. Sla het programma op. Vervolgens gebruik je de terminal om de "executable-bit" te zetten op je programma: `chmod +x programma.py`. Vanaf nu kun je je programma draaien net als een C++ programma: `./programma.py`.

Functies en Files

7. *Zie ook Programmeermethoden opgavenbundel 2.a.* Schrijf een functie die een gegeven temperatuur in graden Fahrenheit omrekent in graden Celsius. Gebruik hiervoor

$$\text{Temp(in Celsius)} = \frac{5}{9} \times (\text{Temp(in Fahrenheit)} - 32)$$

   Vervolgens gebruik je deze functie om een temperatuurschaal af te drukken (maak gebruik van een `for`-loop!):

```
Graden F:    0.0  20.0  40.0  60.0  80.0 100.0 120.0
Graden C:  -17.8  -6.7   4.4  15.6  26.7  37.8  48.9
```

8. *Zie ook Programmeermethoden opgavenbundel 26.* Schrijf een functie `fibo(n)` die het n-de getal van Fibonacci `fibo(n)` berekent. Er geldt `fibo(1) = fibo(2) = 1` en `fibo(n) = fibo(n-1) + fibo(n-2)`, voor $n > 2$. Gebruik geen recursie maar maak gebruik van een lijst en `.append`.

9. Schrijf een functie `def afhalen(lijst, n)` die in een gegeven lijst de `n` grootste getallen opzoekt, deze in een nieuwe lijst plaatst en uit de originele lijst verwijdert. De nieuwe lijst is de return-waarde van de functie. *Tip: Je mag gebruik maken van de functie* `max`*, bijvoorbeeld* `max(lijst)` *geeft de maximumwaarde gevonden in de gegeven lijst.*

   Gebruik deze functie om in de lijst

```
l = [74, 48, 98, 49, 84, 58, 16, 34, 91, 3, 43, 33, 40, 30, 10]
```

   de 3 grootste getallen te vinden en deze achteraan de lijst te plaatsen. Om getallen achteraan de lijst te plaatsen mag je zelf kiezen wat je wilt gebruiken: slice-assignment, de `.extend` methode of de +-operator toegepast op lijsten (of zelf een functie schrijven).

10. Schrijf een functie `def sorteer(lijst):` die de gegeven lijst sorteert met Bubblesort. *Tip: je kunt makkelijk omdraaien met* `lijst[j+1], lijst[j] = lijst[j], lijst[j+1]`.

11. Maak als eerste een *.txt*-bestand waarin op elke regel twee getallen staan gescheiden door een spatie, bijvoorbeeld:

```
65 12
23 42
9 3
43 4
13 95
```

Schrijf vervolgens een programma dat dit bestand regel-voor-regel inleest en voor elke regel het product van de getallen berekent en op het scherm zet.

Als tweede stap kun je het programma uitbreiden door alle producten op te slaan in een lijst en vervolgens deze lijst te sorteren met de functie geschreven in de vorige opgave.

NumPy en matplotlib

12. Deze opgave mag je maken met Python of iPython (voor de "Pylab" modus gebruik `ipython --pylab`). Schrijf expressies om de volgende arrays te maken. Probeer tot een zo kort mogelijke expressie te komen. For-loops gebruiken is **niet** toegestaan. Ook mag je geen toekenningen doen aan individuele array elementen (maar wel aan slices).

```
(a) [ 1  4  7 10 13 16 19 22 25 28]

(b) [4 4 4 4 4 4]

(c) [1 2 3 1 2 3 1 2 3]

(d) [[ 1.  0.  0.  0.]
     [ 0.  1.  0.  0.]
     [ 0.  0.  1.  0.]
     [ 0.  0.  0.  1.]]

(e) [[ 9.  0.  0.]
     [ 0.  9.  0.]
     [ 0.  0.  9.]]

(f) [[1 2 3]
     [1 2 3]
     [1 2 3]]

(g) [[ 0.  0.  0.  0.  0.]
     [ 0.  0.  1.  0.  0.]
     [ 0.  1.  2.  3.  4.]
     [ 0.  0.  3.  0.  0.]
     [ 0.  0.  4.  0.  0.]]
```

Hints: (c) en (f): je mag ook een lijst of array van elementen als waarde geven aan `np.tile()`*, (g): je mag toekenningen aan slices gebruiken.*

13. Deze opgave mag je maken met Python of iPython. Definieer de volgende array:

```
A = np.arange(25).reshape(5, 5)
```

Schrijf nu voor elk van de volgende slices van `A` de bijbehorende slice-expressie:

```
(a) 13

(b) [0 1 2 3 4]

(c) [1 3]

(d) [[ 0  4]
     [ 5  9]
     [10 14]
     [15 19]
     [20 24]]

(e) [[ 6  7  8]
     [11 12 13]
     [16 17 18]]

(f) [[ 6  8]
     [16 18]]
```

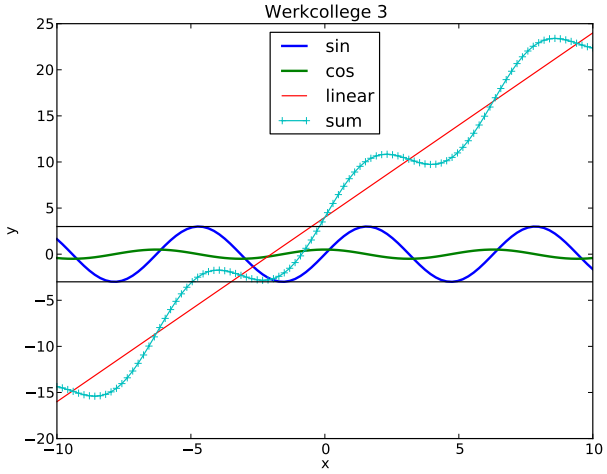
14. Schrijf een programma dat een plot maakt van de volgende functies op een interval $[-10, 10]$ met $N = 100$ stappen (hint: gebruik `np.linspace`):

$$\begin{aligned} y_1 &= 3\sin(x) \\ y_2 &= \frac{1}{2}\cos(x) \\ y_3 &= 2x + 4 \\ y_4 &= y_1 + y_3 \end{aligned}$$

Zorg ervoor dat je de verschillende lijnen goed kunt onderscheiden: maak gebruik van verschillende kleuren, markers, lijndiktes, enz. Geef ook elke lijn een label zodat je een legenda kunt maken.

Plot ook twee **zwarte** lijnen die het bereik van y_1 (voor het interval waarvoor y_1 is geplot) laten zien. *Hint: maak een array met behulp van* `np.tile` *met als parameters de uitkomst van* `np.amin` *en de waarde* N.

Maak de plot netjes op: geef het een titel, plaats aslabels en een legenda. De plot mag of op het scherm worden getoond, of naar een bestand worden geschreven. Voorbeeld:



15. Definieer de volgende array:

```
A = np.arange(125).reshape((5, 5, 5))
```

en bepaal:

(a) Het gemiddelde van elke rij in het laatste "vlak":

```
[ 110.  111.  112.  113.  114.]
```

(b) Voor elk vlak, bepaal de som van elke rij resulterend in een kolom per vlak:

```
[[ 10  35  60  85 110]
 [135 160 185 210 235]
 [260 285 310 335 360]
 [385 410 435 460 485]
 [510 535 560 585 610]]
```

16. Gebruik de module `fractions` om verschillende berekeningen met breuken uit te voeren. Bijvoorbeeld:

(a) $\frac{1}{2} \times 6$
(b) $\frac{4}{9} + \frac{89}{6}$
(c) $\frac{1}{12} + (3 \times \frac{1}{23})$
(d) $3 \div \frac{2}{9}$