Programmeren voor Taal en Tekst 2011: Python

Erik Tjong Kim Sang

23 mei 2011

Onderwerpen

- Week 1: basisdatastructuren en controlstatements (`for, if, while`)
- Week 2: verwerking van strings en reguliere expressies (`sub, search`)
- Week 3: complexe datastructuren (lijsten, dictionaries) en functies
- **Week 4**: werken met bestanden; foutmeldingen, testen

Aanvulling op week 2: Klassen

In Python kunnen klassen met methods (functies) worden gedefinieerd:

```
from random import randrange
MAX = 6
class dobbelsteen:                    # klassedefinitie
   def __init__(self): self.aantal = 0 # initialisatiefunctie
   def gooi(self):                    # functie gooi()
      self.aantal = 1+randrange(MAX)  # kies willekeurig getal
      return(self.aantal)             # geef resultaat terug
def main():
   steen = dobbelsteen()
   print(steen.gooi())
main()
```


WERKEN MET BESTANDEN

Openen van bestanden

Hier is een voorbeeld van bestandsverwerking in Python:

```
bestand = open("bestand.txt","r")   # open bestand
for regel in bestand.readlines():   # voor elke regel
   regel = regel.rstrip()           #    verwijder newline
   print(regel)                     #    print regel
bestand.close()                     # sluit bestand
```

Met de instructie `readlines()` worden alle regels van het bestand gelezen en in een lijst geplaatst

Betere verwerking van bestanden

Meteen alle regels uit een bestand lezen, kost tijd en is soms niet nodig. Hier is een goed alternatief voor de code op de vorige slide:

```
bestand = open("bestand.txt","r")   # open bestand
for regel in bestand:               # voor elke regel
   regel = regel.rstrip()           #    verwijder newline
   print(regel)                     #    print regel
bestand.close()                     # sluit bestand
```

`for regel in bestand` leest steeds 1 regel uit het bestand

Met `break` kan je eventueel de loop eerder beëindigen

Commando's voor bestandsverwerking

`open(bestand,mode)`	open een bestand (mode ={r,w,a})
`bestand.close()`	sluit een bestand
`bestand.read()`	lees een heel bestand, resultaat: string
`bestand.readlines()`	lees een heel bestand, resultaat: lijst
`bestand.readline()`	lees de volgende regel uit een bestand
`print("abc",file=bestand)`	schrijf string "abc" naar bestand

Werken met pipes

Een pipe is een reeks van Unixcommando's die invoer accepteren en of uitvoer genereren. Voorbeeld:

```
import pipes                        # laad pakket pipes
c = "gunzip -c 2011050912.txt.gz" # pipe-commando
p = pipes.Template()                # maak een pipe-object
p.prepend(c,".-")                   # plaats commando in object
bestand = p.open("","r")            # open de pipe
for regel in bestand:               # lees regel uit pipe
   regel = regel.rstrip()           #    verwijder newline
   print(regel)                     #    print regel
bestand.close()                     # sluit pipe
```

Bestanden opvragen uit een directory

Het commando `listdir()` uit het pakket `os` kan worden gebruikt om de namen van bestanden in een bepaalde directory op te vragen:

```
>>> import os
>>> os.listdir()      # vraag bestanden op van deze directory
['Makefile', 'python.png', 's110523.tex']
>>> os.listdir("..")  # vraag bestanden op van ouderdirectory
['cl10', 'pt11']
```

Zonder argument verwerkt de instructie de huidige directory

De naam van de huidige directory vraag je op met `os.getcwd()`

Codering van tekstbestanden

Tekens in tekstbestanden kunnen zijn opgeslagen in verschillende formaten:

- ASCII: 7-bitscode waarin bijna alle standaardtekens passen
- ISO 8859-1: 8-bitscode met extra letters met accenten
- Unicode: 32-bitscode met vele internationale tekens

Python gebruikt standaard Unicode (UTF-8)

Gebruik van bijzondere tekens

Python heeft twee functies beschikbaar voor het omzetten van letters in hun onderliggende code en omgekeerd:

```
>>> ord("Q")    # vraag code van de letter "Q" op
81
>>> chr(82)     # geef letter met code 82
'R'
```

De functie `chr()` kan worden gebruikt om bijzondere tekens op te nemen in een string

Bijvoorbeeld: `chr(8364)` levert het symbool π op

Voorbeeld bijzondere tekens

Stel: je wil vanuit Python "övergångsställe" op het scherm schrijven:

Zoek dan de codes van de bijzondere letters op in een tabel en bouw met `chr()` een string daarmee:

```
>>> chr(0xf6)+"verg"+chr(0o345)+"ngsst"+chr(228)+"lle"
'övergångsställe'
```

Bij functie `ch()` kan je decimale (228), hexadecimale (0xf6) en octale getallen (0o345) gebruiken

FOUTEN EN TESTEN

Het behandelen van runtime-errors: try - except

In Python kan je net als in Java runtime-errors afhandelen in een programma met `try` en `except`:

```
import sys

def main():
   try:
      a,b = input("tik twee getallen in: ").split()
      print("a gedeeld door b is:",eval(a)/eval(b))
   except ZeroDivisionError: print("fout: deling door nul")
   except: print("fout:",sys.exc_info()[0])

main()
```


Het draaien van Pythonprogramma's

Pythonprogramma's kunnen op 3 manieren worden gestart:

```
$ python3 tryexcept        # methode 1
tik twee getallen in: 1 2
a gedeeld door b is: 0.5

$ python3
>>> import tryexcept       # methode 2
tik twee getallen in: 2 4
a gedeeld door b is: 0.5
>>> tryexcept.main()       # methode 3 (mits main() aanwezig)
tik twee getallen in: 3 0
fout: deling door nul
```


Testinstructies voor Pythonpakketten

Een Pythonpakket eindigt meestal met de regel:

```
if __name__ == '__main__': main()
```

Het effect is dat bij het aanroepen op manier 1 `main()` wel wordt uitgevoerd maar bij het aanroepen op manier 2 niet

Bij pakketten bevat `main()` meestal code om de andere functies te testen

Deze code hoeft niet te worden gedraaid als een programma het pakket laadt

Het pakket doctest

Je kan instructies toevoegen aan een Pythonprogramma om de werking ervan te testen:

```
"""             # """: start van doctestinstructies
>>> plus1(6)    # >>>: testinstructie voor doctest
7               #      verwachte uitvoer van testinstructie
"""             # """: einde doctestinstructies

def plus1(a): return(a+1)  # een functie van het programma
```

Vervolgens kan dit programma worden getest met:
`python3 -m doctest programma.py`

Uitvoer van doctest

```
$ python3 -m doctest programma.py  # geen uitvoer: alles ok
$ python3 -m doctest programma.py  # uitvoer bij: return(a+2)
**********************************************************
File "programma.py", line 2, in test
Failed example:
    plus1(6)
Expected:
    7
Got:
    8
**********************************************************
```


Foutmeldingen van Python

Het volgende programma kan leiden tot verschillende foutmeldingen:

```
for a in [0]:
    b = 1
    print(c)
```

Hier is een voorbeeld van zo'n foutmelding:

```
$ python3 error.py
  File "error.py", line 3
    print(c)
          ^
TabError: inconsistent use of tabs and spaces in indentation
```


Verwerken van foutmeldingen

```
for a in [0]:
    b = 1
    print(c)
```

TabError: inconsistent use of tabs and spaces in indentation
: gebruik alleen tabs of alleen spaties bij het inspringen

IndentationError: unexpected indent
: code op hetzelfde niveau moet even ver worden ingesprongen

NameError: name 'c' is not defined
: variabelen moeten een waarde krijgen voordat ze worden gebruikt

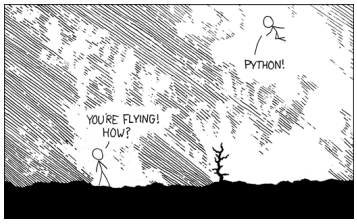
Programmeertips voor beter testen

- gebruik functies voor goedgedefinieerde subtaken
- gebruik geen globaal gedefinieerde vairabelen (behalve constanten) maar parameters voor data-overdracht naar functies
- geef data terug van functies met behulp van `return()`
- test de functies met `doctest`, in `main()` of in een extra zelfgeschreven testfunctie
- test altijd extreme waarden: laagste en hoogste

Achtergrondliteratuur

John Zelle, *Python Programming*. Franklin, Beedle & Associates, tweede editie, 2010, secties 5.4.1, 5.9, 7.1.3, 7.4, 10.3

Unicode HOWTO: `http://docs.python.org/release/3.0.1/howto/unicode.html`
Interface to shell pipelines:
`http://docs.python.org/py3k/library/pipes.html`
Miscellaneous operating system interfaces:
`http://docs.python.org/py3k/library/os.html`
Test interactive Python examples:
`http://docs.python.org/py3k/library/doctest.html`



Bron: http://xkcd.com/533/

THE END