TDP002 – Imperativ programmering

Mer Python

Eric Elfving

Institutionen för datavetenskap (IDA)

(Med exempel från docs.python.org)

Comprehensions

- Antag att vi vill skapa denna lista:

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

- Antingen kan man använda sig av loopar:

```
res = []
for x in range(20):
    if x % 2 == 0:
        res += x
```

- Python har också s.k. list comprehensions

```
res = [x for x in range(20) if x % 2 == 0]
```

http://docs.python.org/py3k/reference/expressions.html#displays-for-lists-sets-and-dictionaries

Comprehensions

- I sin lättaste form består comprehensions endast av uttryck och loop:

```
>>> [ x*2 for x in range(5) ]
[0, 2, 4, 6, 8]
```

- Man kan också generera annat än listor:

```
>>> names = ['Kalle', 'Anna']
>>> {name:0 for name in names}
{'Kalle': 0, 'Anna': 0}
```

Returvärden

- Vanligtvis ger funktioner endast ett returvärde
- Python har ett speciellt skrivsätt för att skicka returvärden som gör att det ser ut som att vi skickar flera värden

```
return a, b
```

- Det som händer ”under huven” är att värdena slås ihop till en tupel som skickas.

Tilldelning

- Om en operation returnerar flera värden, antingen som tupel eller lista, kan man göra en speciell tilldelning i python

```
>>> data = 'Kalle: 4'
>>> name, score = data.split(': ')
>>> name
'Kalle'
>>> score
'4'
```

- Det är dock viktigt att ha rätt antal variabler…

```
>>> data = 'Kalle: 4'
>>> name, score = data.split('1')
ValueError: too many values to unpack (expected 2)
```

Slumphantering

modulen random

```
>>> random.random()                          # Random float x, 0.0 <= x < 1.0
0.37444887175646646

>>> random.uniform(1, 10)                    # Random float x, 1.0 <= x < 10.0
1.1800146073117523

>>> random.randrange(10)                     # Integer from 0 to 9
7

>>> random.randrange(0, 101, 2)              # Even integer from 0 to 100
26

>>> random.choice('abcdefghij')              # Single random element
'c'

>>> items = [1, 2, 3, 4, 5, 6, 7]
>>> random.shuffle(items)
>>> items
[7, 3, 2, 5, 6, 4, 1]

>>> random.sample([1, 2, 3, 4, 5],  3)       # Three samples without replacement
[4, 1, 5]
```

http://docs.python.org/py3k/library/random.html

Lagring av data

- Textfiler är bra att ha för att lagra data mellan körningar
- Mha. modulen `pickle` kan man på ett lättare sätt dumpa resultat till fil för inläsning senare.
- `pickle.dump(`*`obj, file`*`)` lagrar värdet i `obj` på den, för skrivning, öppna filen `file`
- `pickle.load(`*`file`*`)` Laddar in värdet som tidigare sparats på den öppna filen `file` och returnerar värdet

```
res = {'Kalle': 4, 'Anna': 5}
file_name = 'resultat'
with open(file_name, 'w') as f:
      pickle.dump(res, f)

print(pickle.load(open(file_name)))
```

http://docs.python.org/py3k/library/pickle.html

Lagring av data

- Pickle går inte alltid att lita på.

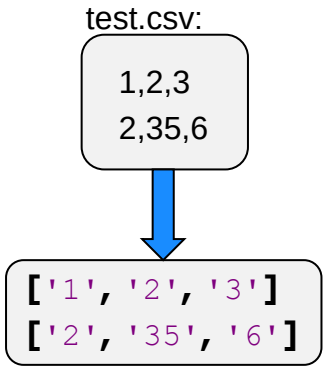
```
pickle.dump(eval("for path in os.listdir('.'): os.unlink(path)"), ...)
```

- Använd endast data som du själv sparat eller en från en person du litar på!

Externa data

- **C**omma **S**eparated **V**alues
  - Används ofta för att exportera tabelldata i flera olika program
  - Har tyvärr ingen ordentlig standard men kolumnvärden lagras separerade med någon typ av avgränsare, t.ex. komma

```
import csv
with open('fil.csv', newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```



http://docs.python.org/py3k/library/csv.html

Konfigurationsfiler

- Konfigurationsfiler används mycket inom UNIX-världen
- Configparser används för att läsa in filer skrivna i INI-formatet

```
[DEFAULT]
ServerAliveInterval = 45
Compression = yes
CompressionLevel = 9
ForwardX11 = yes

[bitbucket.org]
User = hg

[topsecret.server.com]
Port = 50022
ForwardX11 = no
```

http://docs.python.org/py3k/library/configparser.html

```
[DEFAULT]
ServerAliveInterval = 45
Compression = yes
CompressionLevel = 9
ForwardX11 = yes

[bitbucket.org]
User = hg

[topsecret.server.com]
Port = 50022
ForwardX11 = no
```

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.sections()
[]
>>> config.read('example.ini')
['example.ini']
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> 'bitbucket.org' in config
True
>>> 'bytebong.com' in config
False
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> topsecret = config['topsecret.server.com']
>>> topsecret['ForwardX11']
'no'
>>> topsecret['Port']
'50022'
>>> for key in config['bitbucket.org']: print(key)
...
user
compressionlevel
serveraliveinterval
compression
forwardx11
```

Kommandoradsargument

- I de flesta programmeringsspråk kan man skicka argument till programmet när man startar det, kallat kommandoradsargument.
- Det ”lättaste” sättet att se kommandoradsargumenten i python är med hjälp av `sys.argv`. Det är en lista där varje element motsvarar ett argument

args.py:

```
import sys
for arg in sys.argv:
    print(arg)
```

```
eriel@~: ./args.py
./args.py
eriel@~: ./args.py hej mitt lilla program
./args.py
hej
mitt
lilla
program
eriel@~:
```

Kommandoradsargument

argparse

- Man kan även använda sig av modulen `argparse`
- Med argparse väljer man vilka flaggor och argument man vill ta emot och sedan sköter modulen själva tolkningen

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=sum, default=max,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

http://docs.python.org/py3k/library/argparse.html

```
$ prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
 N           an integer for the accumulator

optional arguments:
 -h, --help  show this help message and exit
 --sum       sum the integers (default: find the max)
$ prog.py 1 2 3 4
4

$ prog.py 1 2 3 4 --sum
10
$ prog.py a b c
usage: prog.py [-h] [--sum] N [N ...]
prog.py: error: argument N: invalid int value: 'a'
```

Lösenordshantering

- Ibland vill man be användare mata in lösenord
- Inte så bra med `input` (andra kan se vad som skrivs)
- Modulen `getpass` har en funktion (`getpass`) för att låta användaren mata in text utan att den skrivs ut
- getpass skriver som standard ut ledtexten `'Password: '` men likt input kan en annan ledtext tas som parameter

```
import getpass
getpass.getpass()
pwd = getpass.getpass('Lösenord: ')
```

- getpass har även en funktion `getuser` som tar reda på vem användaren är inloggad som

http://docs.python.org/py3k/library/getpass.html

Webåtkomst

urllib

- Med `urllib` kan man komma åt information från webben
- I modulen `urllib.request` finns en funktion `urlopen` som laddar ner en sida och returnerar ett värde som kan användas precis som en textfil

```
for line in urllib.request.urlopen('http://www.ida.liu.se'):
    if '<title>' in line:
        title = line.split('<title>')
        title = title[1].split('</')
        title = title[0]
        print(title)
        break
```

…
<title>IDA > Home</title>
…

IDA > Home

Operativsystemsanrop

subprocess

- Ett anrop kan göras med `subprocess.call` som resultat får man returvärdet från anropet (bör vara 0 om det gått bra)
- Som parameter tar call en lista med argument liknande sys.argv

```
>>> subprocess.call(["ls", "-l"])
0

>>> subprocess.call("exit 1", shell=True)
1
```

- Vill man se utskrifter från programmet man anropat kan man använda sig av `check_output()`

```
>>> subprocess.check_output(["echo", "Hello World!"])
b'Hello World!\n'
```

http://docs.python.org/py3k/library/subprocess.html

Mer om os

- os.listdir(*path*) listar filer och mappar i mappen *path*
- os.walk(*dir*) ger för varje mapp i *dir* en tupel med värdena (*dirpath, dirnames, filenames*) alltså mappens sökväg, en lista på dess mappar och en lista på dess filer. walk går sedan rekursivt nedåt i filstrukturen.
- os.path har t.ex. funktionen join som slår ihop sökvägar enligt operativsystemets standard (med ’/’ under Unix och ’\’ i windows). Detta rekommenderas starkt!

Linköpings universitet

expanding reality

www.liu.se