Playing around

Comprehensions

definition

filtering

Multiple Values

Examples

References

Comprehensions

Rudiments of Functional Programming

Walter Cazzola

Dipartimento di Informatica
Università degli Studi di Milano
e-mail: cazzola@di.unimi.it
twitter: @w_cazzola

Playing around with ...
Implementing the LS Command

https://docs.python.org/3/library/index.html

```python
import os, sys, time, humanize
from stat import *

modes = {’r’: (S_IRUSR,S_IRGRP,S_IROTH), ’w’: (S_IWUSR,S_IWGRP,S_IWOTH), ’x’: (S_IXUSR,S_IXGRP,S_IXOTH)}

def format_mode(mode):
   s = ’d’ if S_ISDIR(mode) else "-"
   for i in range(3):
      for j in [’r’,’w’,’x’]:
         s += j if S_IMODE(mode) & modes[j][i] else ’-’
   return s

def format_date(date):
   d = time.localtime(date)
   return "{0:4}-{1:02d}-{2:02d} {3:02d}:{4:02d}:{5:02d}".format(
      d.tm_year, d.tm_mon, d.tm_mday, d.tm_hour, d.tm_min, d.tm_sec)

def ls(dir):
   print("List of {0}:".format(dir))
   for file in os.listdir(dir):
      metadata = os.stat(file)
      print("{2} {1:6} {3} {0} ".format(file, approximate_size(metadata.st_size, False), \
         format_mode(metadata.st_mode), format_date(metadata.st_mtime)))

if __name__ == "__main__": ls(sys.argv[1])
```

```
[11:35]cazzola@hymir:~/esercizi-pa>python3 ls-l.py ../esercizi-pa/
List of /home/cazzola/esercizi-pa:
-rw-r--r-- 0.7 KB 2009-10-01 16:00:42 humanize.py
-rw-r--r-- 0.2 KB 2009-10-14 14:30:06 fibonacci.py
-rw-r--r-- 0.9 KB 2009-10-22 11:35:12 ls-l.py
drwxr-xr-x 0.1 KB 2009-10-19 15:14:19 modules
-rw------- 0.2 KB 2009-10-14 11:30:52 factorial.py
-rw-r--r-- 0.3 KB 2009-10-14 16:16:26 ifibonacci.py
```

Comprehensions

Introduction

Comprehensions are a compact way to transform a set of data into another

- it applies to mostly all python's structured type, i.e., lists, sets, dictionaries
- it is in contrast to list all the elements

Some basic comprehensions applied to lists, sets and dictionaries respectively

- a list composed of the first ten integers

```
>>> [elem for elem in range(1,11)]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- a set composed of the first ten even integers

```
>>> {elem*2 for elem in range(1,11)}
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

- a dictionary composed of the first ten couples «n, n^2»

```
>>> {elem:elem**2 for elem in range(1,10)}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Comprehensions

To Filter out Elements of a Dataset

Comprehensions can reduce the elements in the dataset after a constraint.

E.g., to select perfect squares out of the first 100 integers

```
>>> [elem for elem in range(1,101) if (int(elem**.5))**2 == elem]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- range(1,101) generates a list of the first 100 integers (first extreme included, second excluded);
- the comprehension skims through the list selecting those elements whose square of the integral part of its square roots are equal to the element itself

E.g., to select the odd numbers out of a tuple

```
>>> {x for x in (1, 22, 31, 23, 10, 11, 11, -1, 34, 76, 778, 10101, 5, 44) if x%2 != 0}
{1, 5, 11, 10101, 23, -1, 31}
```

- note that the second 11 is removed from the set;
- the set does not respect the tuple order (it is not ordered at all).

Comprehensions
To Select Multiple Values

Comprehensions can select multiple values out of the dataset.

E.g., to swap keys and values in the dictionary

```
>>> a_dict = {’a’: 1, ’b’: 2, ’c’: 3}
>>> {value:key for key, value in a_dict.items()}
{1: ’a’, 2: ’b’, 3: ’c’}
```

Comprehensions can select values out of multiple datasets

E.g., to merge two sets in a set of couples

```
>>> english = [’a’,’b’,’c’,’d’,’e’,’f’,’g’,’h’,’i’,’j’,...,’r’,’s’,’t’,’u’,’v’,’w’,’x’,’y’,’z’]
>>> greek = [′α′,′β′,′χ′,′δ′,′ε′,′φ′,′γ′,′η′,′ι′,′ω′,′ϰ′,′λ′,′μ′,′ν′,′ο′,′π′,′ϙ′,′ρ′,′σ′,′τ′,′ϑ′,′υ′,′ϝ′,′ξ′,′ψ′,′ζ′]
>>> [(english[i],greek[i]) for i in range(0,len(english))]
[(’a’, ’α’), (’b’, ’β’), (’c’, ’χ’), (’d’, ’δ’), (’e’, ’ε’), (’f’, ’φ’), (’g’, ’γ’), (’h’, ’η’),
 (’i’, ’ι’), (’j’, ’ω’), (’k’, ’ϰ’), (’l’, ’λ’), (’m’, ’μ’), (’n’, ’ν’), (’o’, ’ο’), (’p’, ’π’),
 (’q’, ’ϙ’), (’r’, ’ρ’), (’s’, ’σ’), (’t’, ’τ’), (’u’, ’ϑ’), (’v’, ’υ’), (’w’, ’ϝ’), (’x’, ’ξ’),
 (’y’, ’ψ’), (’z’, ’ζ’)]
```

E.g., to calculate the Cartesian product

```
>>> {(x,y) for x in range(3) for y in range(5)}
{(0,1),(1,2),(0,0),(2,2),(1,1),(1,4),(0,2),(2,0),(1,3),(2,3),(2,1),(0,4),(2,4),(0,3),(1,0)}
```

Comprehensions

Comprehensions @ Work: Prime Numbers Calculation

Classic approach to the prime numbers calculation

```
def is_prime(x):
    div =2
    while div <= math.sqrt(x):
       if x%div == 0: return False
       else: div += 1
    return True

if __name__ == "__main__":
    primes = []
    for i in range(1, 50):
       if is_prime(i): primes.append(i)
    print(primes)
```

The algorithm again but using comprehensions

```
def is_prime(x):
    div = [elem for elem in range(2,int(math.sqrt(x))+1) if x%elem ==0 ]
    return len(div) == 0

if __name__ == "__main__":
    print([elem for elem in range(1, 50) if is_prime(elem)])
```

```
[8:50]cazzola@hymir:~/esercizi-pa>python3 imp-sieve.py
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
[1:31]cazzola@hymir:~/esercizi-pa>python3 sieve.py
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Comprehensions

Comprehensions @ Work: Quicksort

```
def quicksort(s):
   if len(s) == 0: return []
   else:
     return quicksort([x for x in s[1:] if x < s[0]]) + \
            [s[0]] +\
            quicksort([x for x in s[1:] if x >= s[0]])

if __name__ == "__main__":
   print(quicksort([]))
   print(quicksort([2, 4, 1, 3, 5, 8, 6, 7]))
   print(quicksort("pineapple"))
   print(’’.join(quicksort(’pineapple’)))
```

```
[23:22]cazzola@hymir:~/esercizi-pa>python3 quicksort.py
[]
[1, 2, 3, 4, 5, 6, 7, 8]
[’a’, ’e’, ’e’, ’i’, ’l’, ’n’, ’p’, ’p’, ’p’]
aeeilnppp
```

References

- Jennifer Campbell, Paul Gries, Jason Montojo, and Greg Wilson.
  Practical Programming: An Introduction to Computer Science Using Python.
  The Pragmatic Bookshelf, second edition, 2009.
- Mark Lutz.
  Learning Python.
  O'Reilly, third edition, November 2007.
- Mark Pilgrim.
  Dive into Python 3.
  Apress*, 2009.