Programação II

Compreensões

Hugo Pacheco

DCC/FCUP
21/22

Listas por compreensão

- É muito comum construir uma lista partindo de uma outra:
  - selecionando elementos usando uma condição
  - aplicando uma transformação a cada elemento
- E.g., calcular quadrados de números de 1 a 9

```
# utilizando ciclo
sqrs = []
for x in range(1, 10):
    sqrs.append(x**2)

# utilizando ordem superior
sqrs = list(map(lambda x:x**2,range(1,10)))

# utilizando compreensão
sqrs = [x**2 for x in range(1,10)]
```

Listas por compreensão

- Outros exemplos

```
>>> [i**2 for i in [2,3,5,7]]
[4, 9, 25, 49]
>>> [1+x/2 for x in [0, 1, 2]]
[1.0, 1.5, 2.0]
>>> [ord(c) for c in "ABCDEF"]
[65, 66, 67, 68, 69, 70]
>>> [len(s) for s in "As armas e os barões
assinalados".split()]
[2, 5, 1, 2, 6, 11]
```

Listas por compreensão

- Outros exemplos com uma condição

```
#quadrados dos números divisores de 3 inferiores a 10
>>> [x**2 for x in range(10) if x%3==0]
[0, 9, 36, 81]
```

```
# utilizando ordem superior
list(map(lambda x:x**2 \
     ,filter(lambda x:x%3==0,range(10)) \
     ))

>>> [(x,x) for x in range(1,7) if x+x not in {2, 12}]
[(2, 2), (3, 3), (4, 4), (5, 5)]
```

Listas por compreensão

- Testar se um número é primo, i.e., a lista de todos os seus divisores próprios é vazia

```
def primo(n):
    # lista dos divisores próprios
    divs = [d for d in range(2,n) if n%d==0]
    # não é primo se e só se a lista for vazia
    return len(divs)==0
```

Listas por compreensão

- Podemos utilizar compreensões aninhadas, e/ou utilizar várias listas

- E.g., para criar a matriz (lista de listas) da multiplicação

```
>>> [[(i,j) for j in range(1,6)] for i in range(1,6)]
[[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)]
,...
[(5, 1), (5, 2), (5, 3), (5, 4), (5, 5)]]
```

- E.g., para calcular o produto cartesiano de dois conjuntos (como listas) (ordem influencia resultado)

```
>>> [(i,j) for i in range(1,3) for j in "ABCDE"]
[(1, 'A'), (1, 'B'),..., (2, 'D'), (2, 'E')]

>>> [(i,j) for j in "ABCDE" for i in range(1,3) ]
[(1, 'A'), (2, 'A'),..., (1, 'E'), (2, 'E')]
```

Compreensão

- A sintaxe por compreensão funciona para qualquer coleção (listas, dicionários, sets, etc); inspirada na teoria de conjuntos

$$\{x^2 : x \in \{1,\ldots,9\}\}$$

$$\{x^2 : x \in \{1,\ldots,9\}, 2 \mid x\}$$

- Utilizando sets em Python (ordem não interessa)

```
>>> {x**2 for x in range(1,10) }
{64, 1, 4, 36, 9, 16, 49, 81, 25}
>>> {x**2 for x in range(1,10) if x%2==0 }
{16, 64, 4, 36}
```

Compreensão

- No fundo só muda o delimitador

- Quando a ordem interessa ⇒ listas

```
>>> [x**2 for x in range(1,6)]
[1, 4, 9, 16, 25]
>>> [(x,x**2) for x in range(1,6)]
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

- Quando não há repetidos e a ordem não interessa ⇒ sets

```
>>> {x**2 for x in range(1,6)}
{1, 4, 9, 16, 25}
>>> {(x,x**2) for x in range(1,6)}
{(2, 4), (4, 16), (1, 1), (3, 9), (5, 25)}
```

- Quando as chaves são únicas e a ordem não interessa ⇒ dicionários

```
>>> {x : x**2 for x in range(1,6)}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Compreensões

- Quando há repetidos, o tipo de coleção é mais pertinente

```
>>> [x*y for x in [1,2,3] for y in [1,2,3]]
[1, 2, 3, 2, 4, 6, 3, 6, 9]

# elimina repetidos
>>> {x*y for x in [1,2,3] for y in [1,2,3]}
{1, 2, 3, 4, 6, 9}

# elimina repetidos, não os conta
>>> {x*y : 1 for x in [1,2,3] for y in [1,2,3]}
{1:1, 2:1, 3:1, 4:1, 6:1, 9:1}

# contar repetidos
>>> d = {}
>>> for xy in [x*y for x in [1,2,3] for y in [1,2,3]]:
>>>     d[xy] = 1 + d.get(xy,0)
{1: 1, 2: 2, 3: 2, 4: 1, 6: 2, 9: 1}
```

- Conseguimos usar apenas compreensões para o último exemplo?

Sets por compreensão

```
fruits = {'apple', 'mango', 'banana','cherry'}

#map em sets funcional
>>> set(map(len,fruits))
{5,6}
# map em sets por compreensão
>>> {len(f)  for f in fruits }
{5,6}
```

```
#filter em sets funcional
>>> set(filter(lambda f: 'a' in f,fruits))
{'mango', 'apple', 'banana'}
#filter em sets por compreensão
>>> { f for f in fruits if 'a' in f }
{'apple', 'banana', 'mango'}

# set do alfabeto
{chr(i)  for i in range(ord('a'),ord('z'))}

# união de listas de sets
>>> l = [{'a','b','c'},{'c','d','e'},{'b','k'}]
>>> {c for s in l for c in s }
{'b', 'd', 'e', 'a', 'k', 'c'}
```

Dicionários por compreensão

```
>>> fruits = ['apple', 'mango', 'banana','cherry']

# valor = comprimento da palavra
>>> dict = {f:len(f) for f in fruits}
{'apple': 5, 'mango': 5, 'banana': 6, 'cherry': 6}

# valor = índice na lista
>>> f_dict = {f:i for i,f in enumerate(fruits)}
{'apple': 0, 'mango': 1, 'banana': 2, 'cherry': 3}

# inverter um dicionário, remove repetidos
>>> d = {v:k for k,v in dict.items()}
{5: 'mango', 6: 'cherry'}

# conversão de moeda
>>> price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}
>>> {item: value*0.76 for (item, value) in old_price.items()}
{'milk': 0.7752, 'coffee': 1.9, 'bread': 1.9}

#nome com 4 letras e idade abaixo de 40
>>> people = {'jack': 38,'michael': 48,'guido': 33,'john': 57}
{ name:age for name,age in dict.items() if len(name)==4 if age<40 }
{'jack': 38}
```

Exemplo *35 Sonnets*

- Extrai dicionário de poemas da obra *35 Sonnets* de Fernando Pessoa

```
import urllib.request
url = 'http://www.gutenberg.org/files/19978/19978-0.txt'
urllib.request.urlretrieve(url,'35sonnets.txt')
with open('35sonnets.txt','r') as f: ls = f.read().splitlines()
```

```
# testa se linha é numeração romana
def roman(s):
    s = s[:-1] #deixa cair último caracter
    bad = { c for c in s if not c.isupper() }
    return len(s) > 0 and len(bad) == 0

# extrai poema
def poem(lines):
    res = []
    for l in lines:
        if len(l) == 0: break
        else: res.append(l)
    return "\n".join(res)

poemas = { l[:-1] : poem(ls[i+3:]) for i,l in enumerate(ls) if roman(l) }
```

Exemplo *35 Sonnets*

- Conta palavras por poema

```
# normaliza string
def trim(str):
    for c in str:
        if not c.isalpha():
            str = str.replace(c,'')
    return str.lower()

# conta palavras
def conta(str):
    palavras = [ trim(p) for p in str.split()]
    count = {}
    for p in palavras:
        count[p] = 1 + count.get(p,0)
    return count

palavras_por_poema = { i:conta(p) for i,p in poemas.items() }

print(palavras_por_poema)
{'I': {'whether': 1, 'we': 9,...}, 'II': {...}}
```

Exemplo *35 Sonnets*

- Conta palavras em toda a obra

```
# junta dicionários para contar palavras em toda a obra
palavras_total = {}
for ps in palavras_por_poema.values():
    for p,n in ps.items():
        palavras_total[p] = n + palavras_total.get(p,0)

print(palavras_total)
{'whether': 1, 'we': 42, 'write': 1, 'or': 18,...}
```

Exemplo *35 Sonnets*

```
palavras_total = sorted(palavras_total.items(),key=lambda kv: kv[1])
mais_frequentes = dict(palavras_total[-10:])

import matplotlib.pyplot as plt
plt.bar(mais_frequentes.keys(), mais_frequentes.values())
plt.show()
```

- Desenhar um histograma com as 10 palavras mais frequentes (vamos estudar gráficos mais tarde)