Um pouco mais sobre iterações

MC102-2018s1-Aula08-180322-takeaway

Arthur J. Catto, PhD
ajcatto@g.unicamp.br

22 de março de 2018

1 Um pouco mais sobre iterações

1.1 Revisitando `input`

Sabemos que `input` retorna uma *string*, p.ex.

```
In [ ]: s = input('Dados? ')
        print(type(s), repr(s))
```

Os itens nessa *string* podem ser separados por `split` e colocados numa *lista*...

```
In [ ]: s = '1,2.34,5.67,89'

        ss = s.split('.')
        print(type(ss), repr(ss))
        print(repr(s))
```

... mas os itens dessa *lista* continuam sendo *strings*.
Veja o que acontece quando tentamos somar os dois primeiros itens da lista...

```
In [ ]: s = '1  2.34 5.67 89'

        ss = s.split()
        print(ss[0] + ss[1], repr(ss[0] + ss[1]), type(ss[0] + ss[1]))
```

... como esses itens são *strings* eles foram concatenados e não somados...
Podemos usar um `for` para converter *string* em *int* ou *float*...

```
In [ ]: nums = []
        for k in range(4):
            nums.append(k)

In [5]: s = '1  2.34 5.67 89'

        ss = s.split()
        print(ss)
        ssf = []
```

```
        for x in ss:
            ssf += [float(x)]
        print(type(ssf[0] + ssf[1]), ssf[0] + ssf[1])
```

```
['1', '2.34', '5.67', '89']
<class 'float'> 3.34
```

E podemos combinar tudo isso num único comando, se repetirmos com cuidado todos os passos dados...

1. Lemos uma linha de texto: `input('Dados? ')`
2. Separamos os itens nessa linha: `input('Dados? ').split()`
3. Construímos uma lista com os itens separados:
   `[x for x in input('Dados? ').split()]`
4. Como os x são *strings* mas queremos *floats*, corrigimos isso:
   `[float(x) for x in input('Dados? ').split()]`
5. E atribuímos um nome ao resultado:
   `ssf = [float(x) for x in input('Dados? ').split()]`

```
In [6]: ssf = [float(x) for x in input('Dados? ').split()]
        print(ssf)
```

```
Dados? 1  2.34 5.67 89
[1.0, 2.34, 5.67, 89.0]
```

1.2 List comprehensions

No exemplo anterior, usamos informalmente List Comprehensions — um conceito de Python que a definição de listas de modo conciso.

Você certamente já encontrou definições matemáticas como...

- $s = \{x^2 : x\ in \{0...9\}\}$
- $v = (1, 2, 4, 8, ..., 2^{12})$
- $m = \{x \mid x\ \textbf{in}\ s \text{ e } x \text{ é par}\}$

Python permite representá-las como listas, de uma forma muito natural...

```
In [7]: s = [ x ** 2 for x in range(10)]
        print(s)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [8]: t = [ 2 ** k for k in range(13)]
        print(t)
```

```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
```

```
In [9]: u = [x
             for x in s
             if x % 2 == 0]
        print(u)
```

```
[0, 4, 16, 36, 64]
```

Uma *list comprehension* tem a forma geral

```
vals = [expression
        for value in collection
        if condition]
```

e é equivalente à seguinte sequência de comandos

```
vals = []
for value in collection:
    if condition:
        vals.append(expression)
```

```
In [10]: v = [3 * x for x in u]
         print(v)
```

```
[0, 12, 48, 108, 192]
```

```
In [2]: pals = "the quick brown fox jumps over the lazy dog".split()
        print(pals)
```

```
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

```
In [4]: for i in range(0, len(pals)):
            print(pals[i], end=' ')

        print('\n')

        for pal in pals:
            print(pal, end=' ')
        print()
```

```
the quick brown fox jumps over the lazy dog

the quick brown fox jumps over the lazy dog
```

```
In [12]: prim_letras = [p[0] for p in pals]
         print(prim_letras)
```

```
['t', 'q', 'b', 'f', 'j', 'o', 't', 'l', 'd']
```

```
In [17]: print(pals)
         pals_mod = [[x.upper(), x.lower(), len(x)] for x in pals]
         for x in pals_mod:
             print(x)
```

```
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
['THE', 'the', 3]
['QUICK', 'quick', 5]
['BROWN', 'brown', 5]
['FOX', 'fox', 3]
['JUMPS', 'jumps', 5]
['OVER', 'over', 4]
['THE', 'the', 3]
['LAZY', 'lazy', 4]
['DOG', 'dog', 3]
```

```
In [19]: texto = "letras123-_45+=símbolos67e números 89misturados 0"
         nums = [int(c)
                 for c in texto
                 if c.isdigit()]
         print(nums)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```

Na aula passada, linearizamos uma lista de listas usando o código abaixo...

```
In [21]: llista = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]
         llin = []
         for x in llista:
             llin += x
         print(llin)
         print(llista)
```

```
[11, 12, 13, 21, 22, 23, 31, 32, 33]
[[11, 12, 13], [21, 22, 23], [31, 32, 33]]
```

O mesmo resultado pode ser obtido com uma *list comprehension*...

```
In [22]: llista = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]
         llin = [x
                 for y in llista
                 for x in y]
         print(llin)
```

```
[11, 12, 13, 21, 22, 23, 31, 32, 33]
```

1.2.1 Exemplo: *Dada uma linha de texto contendo inteiros não-negativos, exibir o maior ímpar dentre eles*

Já vimos que uma solução para um problema semelhante pode ser expressa como uma sequência de três ações:

- Ler todos os candidatos
- Encontrar o maior número ímpar dentre os candidatos lidos
- Exibir o resultado ou uma mensagem apropriada caso todos os candidatos sejam pares.

```
In [4]: # Ler todos os candidatos
        cands = [int(x) for x in input('Dados? ').split()]
        print(cands)
```

```
Dados? 12 24 54 64 78 98
[12, 24, 54, 64, 78, 98]
```

```
In [2]: # Encontrar o maior número ímpar dentre os candidatos lidos
        maior_impar = -1
        for cand in cands:
            if (cand % 2 == 1) and (cand > maior_impar):
                maior_impar = cand
```

```
In [3]: # Exibir o resultado ou uma mensagem de erro apropriada
        if maior_impar == -1:
            print("Nenhum candidato ímpar.")
        else:
            print("maior ímpar =", maior_impar)
```

```
maior ímpar = 89
```

É possível obter o mesmo resultado reescrevendo os dois últimos blocos para aproveitar a simplicidade de *list comprehensions*...

```
In [6]: # Separar os ímpares dentre os candidatos lidos
        impares = [x for x in cands if x % 2 == 1]
        print(impares)
```

```
[]
```

```
In [7]: if len(impares) == 0:
            print("Nenhum candidato ímpar.")
        else:
            print("maior ímpar =", max(impares))
```

```
Nenhum candidato ímpar.
```

1.3 Outros objetos iteráveis

Há vários tipos de objetos iteráveis que podem ser usados num **for**. Por exemplo ...

- listas
- *ranges*
- cadeias de caracteres (*strings*)
- conjuntos
- tuplas

Por exemplo, examine os códigos abaixo e tente prever o resultado dos `prints`...

```
In [41]: for x in range(10, 1, -3):    # aqui o objeto iterável é uma range
             print(x, end=' ')
         print()

10 7 4


In [43]: for x in 'carranca':    # aqui o objeto iterável é uma string
             print(x, end='-')
         print()

c-a-r-r-a-n-c-a-


In [24]: lista = list('carranca')   # aqui o objeto iterável é uma lista
         for x in lista:
             print(x, end='-')
         print()

c-a-r-r-a-n-c-a-


In [25]: for x in {'c', 'a', 'r', 'r', 'a', 'n', 'c', 'a'}:    # aqui o objeto iterável é um con
             print(x, end='-')
         print()

n-r-a-c-


In [26]: conj = set('carranca')
         for x in conj:    # aqui o iterador é um conjunto
             print(x, end='-')
         print()

n-r-a-c-
```

1.3.1 Exercício rápido

Substitua o comentário no código abaixo por um comando **for**.

```
In [ ]: numXs = int(input('Quantos X eu devo imprimir? '))
        # imprimir numXs Xs
```

Solução

```
In [10]: numXs = int(input('Quantos X eu devo imprimir? '))
         # imprimir numXs Xs
         for _ in range(numXs):
             print('X', end='')
         print()
```

```
Quantos X eu devo imprimir? 5
XXXXX
```

1.3.2 Exercício rápido

Substitua o comentário no código abaixo pelos comandos necessários, incluindo um **for**

```
In [ ]: palavra = input('Digite uma palavra qualquer: ')

        # contar o número de vogais e consoantes em palavra

        print(palavra, 'tem', n_caracteres, 'caracters,', end=' ')
        print('incluindo', n_vogais, 'vogais e', n_consoantes, 'consoantes.')
```

Solução 1

```
In [29]: texto = input('Digite uma linha de texto qualquer: ')
         n_caracteres = len(texto)
         n_vogais = 0
         n_consoantes = 0
         for c in texto:
             if c in 'aáàãâeéêiíoóõôuú':
                 n_vogais += 1
             elif c in 'bcçdfghjklmnpqrstvwxyz':
                 n_consoantes += 1
         print(texto, 'tem', n_caracteres, 'caracters,', end=' ')
         print('incluindo', n_vogais, 'vogais e', n_consoantes, 'consoantes.')
```

```
Digite uma linha de texto qualquer: the quick brown fox jumps over the lazy dog
the quick brown fox jumps over the lazy dog tem 43 caracters, incluindo 11 vogais e 24 consoante
```

Solução 2

```
In [28]: texto = input('Digite uma linha de texto qualquer: ')
         vogais = [c for c in texto if c in 'aáàãâeéêiíoóõôuú']
         consoantes = [c for c in texto if c in 'bcçdfghjklmnpqrstvwxyz']
         print(texto, 'tem', len(texto), 'caracters,', end=' ')
         print('incluindo', len(vogais), 'vogais e', len(consoantes), 'consoantes.')
```

```
Digite uma linha de texto qualquer: the quick brown fox jumps over the lazy dog
the quick brown fox jumps over the lazy dog tem 43 caracters, incluindo 11 vogais e 24 consoante
```

1.3.3 Exercício rápido

Ler uma linha de texto com uma sequência de inteiros e exibir a soma dessa sequência.

```
In [ ]: # Ler uma sequência de inteiros
        # Calcular a soma dessa sequência
        # Exibir o resultado

In [ ]: # Ler uma sequência de inteiros de uma linha de texto

In [ ]: # Calcular a soma dessa sequência

In [ ]: # Exibir o resultado
        print("soma da lista =", soma)
```

Solução

```
In [ ]: # Ler uma sequência de inteiros de uma linha de texto
        print("Digite uma sequência de inteiros: ")
        nums = [int(x) for x in input().split()]

In [ ]: # Calcular a soma dessa sequência
        soma = 0
        for x in nums:
            soma += x

In [ ]: # Exibir o resultado
        print("soma da lista =", soma)
```

1.3.4 Exercício

Ler uma linha de texto e, depois, uma palavra e contar quantas vezes essa palavra aparece na linha de texto lida.

Exemplo de teste Linha de texto: 'Onde digo "Digo", não digo "Digo", digo "Diogo".'
Palavra: digo
Resposta: 5

Um esboço de solução poderia ser...

```
In [ ]: # Ler uma linha de texto e, depois, uma palavra
        # Remover a pontuação da linha de texto e convertê-la em minúsculas
        # Separar as palavras da linha de texto e colocá-las numa lista
        # Contar quantas vezes a palavra dada aparece na lista
        # Exibir o resultado
```

Tente expandir cada um dos comentários abaixo para chegar à solução do problema...

```
In [ ]: # Ler uma linha de texto e, depois, uma palavra
```

```
In [ ]: # Remover a pontuação da linha de texto e convertê-la em minúsculas
```

```
In [ ]: # Separar as palavras da linha de texto e colocá-las numa lista
```

```
In [ ]: # Contar quantas vezes a palavra dada aparece na lista
```

```
In [ ]: # Exibir o resultado
```

Solução

```
In [12]: # Ler uma linha de texto e, depois, uma palavra
         texto = input("Texto? ")
         palavra = input("Palavra? ").lower()
```

```
Texto? Onde digo "Digo", não digo "Digo", digo "Diogo".
Palavra? Digo
```

```
In [13]: # Remover a pontuação da linha de texto
         pontuacao = set(',.;:?!"' + "'")
         palavras = ''
         for caracter in texto:
             if caracter in pontuacao:
                 palavras += ' '
             else:
                 palavras += caracter
         print(palavras)
```

```
Onde digo  Digo   não digo  Digo   digo  Diogo
```

```
In [14]: # Converter a linha e a palavra para minúsculas
         palavras = palavras.lower()
         print(palavras)
         palavra = palavra.lower()
         print(palavra)
```

```
onde digo  digo   não digo  digo   digo  diogo
digo
```

```
In [15]: # Separar as palavras da linha de texto e colocá-las numa lista
         palavras = palavras.split()
         palavras
```

```
Out[15]: ['onde', 'digo', 'digo', 'não', 'digo', 'digo', 'digo', 'diogo']
```

```
In [ ]: # Contar quantas vezes a palavra dada aparece na lista
        quantas = 0
        for p in palavras:
            if p == palavra:
                quantas += 1
```

... ok, mas esse não é o único jeito...

```
In [16]: # Contar quantas vezes a palavra dada aparece na lista
         quantas = palavras.count(palavra)
         quantas
```

```
Out[16]: 5
```

```
In [17]: # Exibir o resultado
         print(palavra, 'aparece', quantas, 'vezes')
         print('em', "'" + texto + "'.")
```

```
digo aparece 5 vezes
em 'Onde digo "Digo", não digo "Digo", digo "Diogo".'.
```