Programação I
Aula 12 — Mais sobre cadeias e listas

Pedro Vasconcelos
DCC/FCUP

Nesta aula

Formatação de texto

Por vezes necessitamos de especificar como são mostrados os resultados de uma computação:

- o número de algarismos e casas decimais;
- como ou sem zeros à esquerda/direita;
- colunas alinhadas numa tabela.

As linguagens de programação permitem fazer isto com comando ou bibliotecas para formatação de texto.

Formatação de texto (cont.)

- Há várias formas de fazer formação em Python
- Vamos ver como usar % como operador de formatação
- Compatível com as versões 2 e 3 de Python

Operador de formatação

formato `%` *valores*

- o *formato* é uma cadeia de carateres com campos marcados por carateres `%`
- um ou mais *valores* agregados num tuplo
- o *resultado* é uma cadeia de carateres com os campos substituidos pelos valores correspondentes

Exemplos:

```
>>> "O valor de Pi é %f" % math.pi
'O valor de Pi é 3.141593'
>>> "%d/%d/%d" % (1, 6, 2013)
'1/6/2013'
```


Alguns campos de formatos

%d inteiro com sinal

```
"%d/%3d/%-3d" % (5, 5, 5)
'5/  5/5  '
```

%e, %f, %g vírgula flutuante, formato exponencial ou decimal

```
"%f %.3f %e" % (math.pi,math.pi,math.pi)
'3.141593 3.142 3.141593e+00'
```

%s cadeia de carateres

```
"(%s/%4s/%-4s)" % ("A", "BC", "D")
'(A/  BC/D   )'
```

%% o caracter %

```
"%d%%" % 12
12%
```

Exemplo

Imprimir uma tabela das funções seno e coseno no intervalo $[0, 2\pi]$.

Primeira versão (sem formatação).

```
from math import *
print("x", "sin(x)", "cos(x)")
for i in range(11):
    x = 2*pi/10 * i
    print(x, sin(x), cos(x))
```


Exemplo (cont.)

Resultado:

```
x   sin(x)   cos(x)
0.0 0.0 1.0
0.6283185307179586 0.5877852522924731 0.8090169943749475
1.2566370614359172 0.9510565162951535 0.3090169943749745
1.8849555921538759 0.9510565162951536 -0.3090169943749734
2.5132741228718345 0.5877852522924732 -0.8090169943749473
3.141592653589793 1.2246467991473532e-16 -1.0
3.7699111843077517 -0.587785252292473 -0.8090169943749476
4.39822971502571 -0.9510565162951535 -0.3090169943749756
5.026548245743669 -0.9510565162951536 0.3090169943749723
5.654866776461628 -0.5877852522924734 0.8090169943749473
6.283185307179586 -2.4492935982947064e-16 1.0
```

Exemplo (cont.)

Segunda versão (usando formatação):

```
print('%7s  %7s  %7s' % ('x', 'sin(x)', 'cos(x)'))
for i in range(11):
    x = 2*pi/10 * i
    print('%7.4f  %7.4f  %7.4f' % (x,sin(x),cos(x)))
```

Legenda:

- `%7.4f` campo de vírgula flutuante com 7 carateres no total e 4 casas decimais;
- `%7s` campo de texto com 7 carateres no total.

Exemplo (cont.)

Resultado:

```
      x    sin(x)    cos(x)
 0.0000    0.0000    1.0000
 0.6283    0.5878    0.8090
 1.2566    0.9511    0.3090
 1.8850    0.9511   -0.3090
 2.5133    0.5878   -0.8090
 3.1416    0.0000   -1.0000
 3.7699   -0.5878   -0.8090
 4.3982   -0.9511   -0.3090
 5.0265   -0.9511    0.3090
 5.6549   -0.5878    0.8090
 6.2832   -0.0000    1.0000
```

Tabela de multiplicação

Outro exemplo: imprimir a tabela da multiplicação de 1 a 10.

Tabela de multiplicação (cont.)

```
 *|  1   2   3   4   5   6   7   8   9  10
==========================================
 1|  1   2   3   4   5   6   7   8   9  10
 2|  2   4   6   8  10  12  14  16  18  20
 3|  3   6   9  12  15  18  21  24  27  30
 4|  4   8  12  16  20  24  28  32  36  40
 5|  5  10  15  20  25  30  35  40  45  50
 6|  6  12  18  24  30  36  42  48  54  60
 7|  7  14  21  28  35  42  49  56  63  70
 8|  8  16  24  32  40  48  56  64  72  80
 9|  9  18  27  36  45  54  63  72  81  90
10| 10  20  30  40  50  60  70  80  90 100
```

Tabela de multiplicação (cont.)

```
# Tabela de multiplicação de 1 a 10
# imprimir a linha de cabeçalho
fmt = "  *|" + 10*"%3d "
print(fmt % tuple(range(1,11)))
print(43*"=")

# imprimir 10 linhas do corpo
for i in range(1,11):
    linha = [i] + [i*j for j in range(1,11)]
    fmt = "%2d|" + 10*"%3d "
    print(fmt % tuple(linha))
```


Mais operações sobre cadeias

Seja `txt` uma cadeia de carateres.

`txt.split()` partir a cadeia na lista das partes delimitadas por espaços

`txt.split(sep)` partir a cadeia na lista das partes delimitadas pela cadeia *sep*

`txt.join(lista)` juntar uma lista de cadeias numa só usando *txt* como separador

Exemplos

```
>>> "as armas e os barões".split()
['as', 'armas', 'e', 'os', 'barões']

>>> "abc-de-fgh".split('-')
['abc', 'de', 'fgh']

>>> " ".join(['as', 'armas', 'e', 'os', 'barões'])
'as armas e os barões'

>>> "--".join(['as', 'armas', 'e', 'os', 'barões'])
'as--armas--e--os--barões'
```


Exemplo maior: a cifra de César

Recorde a solução apresentada na aula 8:

```
def cifrar(k: int, txt: str) -> str:
    msg = ""
    for c in txt:
        msg = msg + rodar(k,c)
    return msg
```

Esta solução é correta mas ineficiente:

- acrescenta um carater de cada vez à cadeia `msg`
- isto constroi muitas cadeias intermédias desnecessárias
- podemos evitar isto usando `join`

Exemplo maior: a cifra de César (2)

Uma solução melhor:

```
def cifrar(k: int, txt: str) -> str:
    lista = [rodar(k,c) for c in txt]
    return "".join(lista)
```

- Construimos uma lista com os carateres individuais rodados
- Usamos `join` para juntar todos carateres numa nova cadeia

Mais geralmente

Em vez de

```
txt = ""
for x in lista:
    txt = txt + x
```

devemos usar

```
"".join(lista)
```

Saída e continuação num ciclo

Duas instruções que permitem alterar a execução de um ciclo:

- **break** sair a meio do ciclo
- **continue** passar à próxima iteração

Além disso:

- **return** termina a função e retorna um resultado (também interrompe qualquer ciclo)

Break com um ciclo for

```
for i in [12, 16, 17, 24, 29]:
    if i % 2 == 1:  # se é ímpar
        break         # termina o ciclo
    print(i)
print("fim")
```

Resultado:

```
12
16
fim
```

Continue com um ciclo for

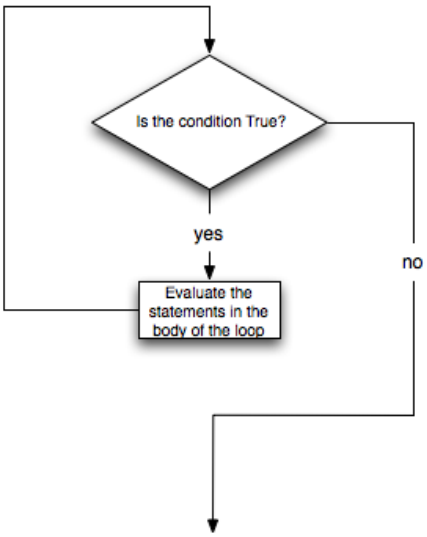
```
for i in [12, 16, 17, 24, 29, 30]:
    if i % 2 == 1:      # se é impar
        continue         # passa ao próximo
    print(i)
print("fim")
```

Resultado:

```
12
16
24
30
fim
```


Saída de um ciclo while

O teste do ciclo *while* ocorre antes da execução do corpo.



Podemos usar *break* para colocar um teste no corpo do ciclo.

Exemplo: teste no meio do corpo

```
total = 0
while True:
    resposta = input("Entre um número (ou vazio)")
    if resposta == '':
        break           # termina o ciclo
    total += int(resposta)
print("Total = ", total)
```


Exemplo: teste no fim do corpo

```
import random
# escolher um inteiro entre [1, 1000]
number = random.randint(1, 1000)
guesses = 0
while True:
    guess = int(input("Número entre 1 e 1000: "))
    guesses += 1
    if guess > number:
        print(guess, "é alto.")
    elif guess < number:
        print(guess, "é baixo.")
    else:
        break
print(guess, "é correto.")
print(guesses, "tentativas.")
```