Disciplina BCM0505-15
Processamento da Informação

Comandos de repetição em Python

Profa. Carla Negri Lintzmayer

carla.negri@ufabc.edu.br

http://professor.ufabc.edu.br/~carla.negri

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Agenda

Comando de repetição “enquanto” em Python

Outros tipos em Python

Comando de repetição “para” em Python

Comando de repetição “enquanto” em Python

Comando “enquanto” em Python

A sintaxe do comando “enquanto” em Python é:

```
1 while expressão lógica:
2     comando(s) executado(s) se expressão for True
3 comando(s) executado(s) após o fim do laço
```

E ele funciona exatamente como vimos antes: teste se a expressão é `True` e, caso seja, execute o corpo do laço; ao fim, teste novamente. Quando a expressão for `False`, o corpo não é executado e passa-se ao próximo comando após o corpo.

Exemplo - Levemente errado

```
1  alice = int(input())
2  bob = int(input())
3  while eh_par(alice + bob):
4      alice = int(input())
5      bob = int(input())
6      print("Alice ganhou!")
7  print("Bob finalmente ganhou!")
```

Exemplo

```
1  alice = int(input())
2  bob = int(input())
3  while eh_par(alice + bob):
4      print("Alice ganhou!")
5      alice = int(input())
6      bob = int(input())
7  print("Bob finalmente ganhou!")
```

Exemplo

```
alice = int(input())
bob = int(input())
while eh_par(alice + bob):
    print("Alice ganhou!")
    alice = int(input())
    bob = int(input())
print("Bob finalmente ganhou!")
```

```
alice_ganhando = True
while alice_ganhando:
    alice = int(input())
    bob = int(input())
    if eh_par(alice + bob):
        print("Alice ganhou!")
    else:
        alice_ganhando = False
print("Bob finalmente ganhou!")
```

Outros tipos em Python

Outros tipos em Python

- Antes de apresentar o comando “para” em Python, precisamos formalmente definir outros tipos básicos de Python.

Outros tipos em Python

- Antes de apresentar o comando “para” em Python, precisamos formalmente definir outros tipos básicos de Python.
- Além do `int`, `float`, `str`, `bool`, Python também oferece `tuple` e `list`.

Outros tipos em Python

- Antes de apresentar o comando “para” em Python, precisamos formalmente definir outros tipos básicos de Python.
- Além do `int`, `float`, `str`, `bool`, Python também oferece `tuple` e `list`.
- Atenção: não começaremos a usar tuplas e listas ainda e, principalmente, não usaremos funções pré-existentes para esses tipos.

Tipo tupla

- É uma sequência ordenada de valores de algum outro tipo.

Tipo tupla

- É uma sequência ordenada de valores de algum outro tipo.
- Para criar uma tupla, devemos saber quantos elementos queremos dentro dela e não será possível adicionar, alterar ou remover seus elementos.

Tipo tupla

- É uma sequência ordenada de valores de algum outro tipo.
- Para criar uma tupla, devemos saber quantos elementos queremos dentro dela e não será possível adicionar, alterar ou remover seus elementos.
- É uma boa opção quando queremos trabalhar com informações diferentes em uma mesma variável.

Tipo tupla

- É uma sequência ordenada de valores de algum outro tipo.
- Para criar uma tupla, devemos saber quantos elementos queremos dentro dela e não será possível adicionar, alterar ou remover seus elementos.
- É uma boa opção quando queremos trabalhar com informações diferentes em uma mesma variável.
- Para criar uma tupla, basta adicionar seus elementos entre parênteses e separá-los por vírgula:

```
identificador = (el1, el2, ..., eln)
```

onde $n \geq 1$.

Tupla

```
1  # leitura das coordenadas de um ponto
2  x = float(input())
3  y = float(input())
4  ponto = (x, y)
5
6  # informações sobre uma pessoa
7  nome = input()
8  idade = int(input())
9  telefone = input()
10 pessoa = (nome, idade, telefone)
```

Tipo lista

- É também uma sequência ordenada de valores de algum outro tipo.

Tipo lista

- É também uma sequência ordenada de valores de algum outro tipo.
- Para criar uma lista, basta adicionar seus elementos entre colchetes e separá-los por vírgula:

```
identificador = [el1, el2, ..., eln]
```

onde $n \geq 1$.

Lista

```
1  vogais = ["a", "e", "i", "o", "u"]
2  print(vogais)
3
4  digitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
5  print(digitos)
```

Operações em listas e tuplas

- Cada valor armazenado em uma lista ou tupla é identificado por um **índice**.

Operações em listas e tuplas

- Cada valor armazenado em uma lista ou tupla é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.

Operações em listas e tuplas

- Cada valor armazenado em uma lista ou tupla é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.

Operações em listas e tuplas

- Cada valor armazenado em uma lista ou tupla é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.
- A função `len` devolve o tamanho de uma lista ou tupla que lhe é passada por parâmetro.

Operações em listas e tuplas

- Cada valor armazenado em uma lista ou tupla é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.
- A função `len` devolve o tamanho de uma lista ou tupla que lhe é passada por parâmetro.
- O operador `in` serve para verificar se um valor está contido em uma lista ou tupla.

Operações em listas e tuplas

```
1  vogais = [ "a", "e", "i", "o", "u" ]
2  print(vogais)
3  print(vogais[2])
4  vogais[3] = "x"
5  print(vogais)
6  print("e" in vogais)
7
8  ponto = (4, 5)
9  print(ponto[0])
10 ponto[0] = 3   # ERRO!!
11 print(6 in ponto)
```

Listas

- Observe que nós já vimos listas!
- Quando temos uma string e usamos a função `split()` sobre ela, o conteúdo devolvido é uma lista.

```
1 entrada = input()
2 entrada = entrada.split()
3 a = int(entrada[0])
4 b = float(entrada[1])
```

Listas numéricas

- Em várias aplicações, é útil criar listas que contêm apenas números.

```
1  primeiros_10 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Listas numéricas

- Em várias aplicações, é útil criar listas que contêm apenas números.

```
1  primeiros_10 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- Python oferece a função `range()`, que não devolve uma lista, mas podemos usá-la em conjunto com a função `list()` para fazer a conversão.

```
1  primeiros_10 = list(range(1,11))
```

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;
- `range(ini, fim)`: cria uma lista com valores entre ini e $fim - 1$, de uma em uma unidade;

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;
- `range(ini, fim)`: cria uma lista com valores entre ini e $fim - 1$, de uma em uma unidade;
- `range(ini, fim, passo)`: cria uma lista com valores entre ini e $fim - 1$, começando em ini e seguindo a cada $passo$ unidades.

Alguns exemplos

```
1  print(list(range(10)))
2
3  print(list(range(3,15)))
4
5  print(list(range(15,3)))
6
7  print(list(range(15,3,-1)))
```

Comando de repetição “para” em Python

Comando “para” em Python

A sintaxe do comando “para” em Python é:

```
1 for var in lista:
2     comando(s) executado(s) para cada elemento de lista
3 comando(s) executado(s) após o fim do laço
```

Ele funciona atribuindo à variável `var` cada valor que está na lista, na ordem, e executando o conteúdo do corpo do laço para cada um deles.

Alguns exemplos

```
1  for i in [1, 2, 3, 4, 5]:
2      print(i)
3
4  n = int(input())
5  for i in range(n):
6      print(i)
7
8  for elem in ['gato', 'cachorro', True, 34]:
9      print(e)
```

Observe

```
1  n = int(input())
2  soma = 0
3  for i in range(1,n+1):
4      soma = soma + i
```

Observe

```python
n = int(input())
soma = 0
for i in range(1,n+1):
    soma = soma + i
```

é idêntico a

1: $\textsc{Leia}(n)$
2: $soma \leftarrow 0$
3: **Para** $i \leftarrow 1$ **até** n, **com** $i \leftarrow i + 1$ **faça**
4: $\quad soma \leftarrow soma + i$

Exemplo

```
1  def fatorial(n: int) -> int:
2      fat = 1
3      for i in range(2,n+1):
4          fat = fat * i
5      return fat
6
7  def main():
8      n = int(input())
9      k = int(input())
10
11     coef_binom = fatorial(n) / (fatorial(k) * fatorial(n-k))
12
13     print("O coeficiente binomial (%d %d) é" % (n, k,
       ↪  coef_binom))
14
15 main()
```