

Funções

Programação I
2017.2018

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Revisões

Funções

Definição e utilização

Porquê usar funções?

Como programar?

Revisões

Iteração

Instrução while

```
while <condicao>:  
    <bloco de instruções>
```

Fluxo de execução

1. Avaliar a condição, obtendo True ou False
2. Se False, sai da instrução while e continua com próxima instrução
3. Se True, executa o corpo do while e volta ao passo 1

Instrução break

Termina o ciclo antecipadamente

Instrução continue

Passa de imediato para nova iteração

fazendo o teste e não executando as instruções restantes do bloco

Cuidados

Verificar o valor das variáveis da condição

Para o ciclo terminar devem ser alteradas no corpo do while

Verificar a condição de paragem

É comum existir uma iteração a mais ou a menos

Verificar efeitos da interrupção antecipada

As instruções break e continue podem produzir efeitos não desejados no comportamento do ciclo

Funções

Função

Sequência de instruções com nome que realiza uma computação

Uma função

Tem um nome

Recebe argumentos

Devolve um resultado

É executada sempre que o seu nome é invocado

Exemplo

`valor = maximo()`

Módulo

Conjunto de funções relacionadas (contidas num ficheiro)

Utilização

```
>>> import math  
>>> print(math)  
<module 'math' from '/usr/lib/python3.5/math.py'>
```

Notação dot

```
>>> degrees = 45  
>>> radian = degrees / 360*2*math.pi  
>>> math.sin(radians)  
0.707106781187
```

Composição de funções

$$f \circ g (x) = f (g(x))$$

Exemplos

```
x = math.sin( degrees / 360*2*math.pi)
```

```
x = math.exp( math.log(x+1) )
```

Definição e utilização

Definição de uma função

Especifica

o nome e parâmetros da função
a sequência de instruções a executar

Possui

Cabeçalho – nome, parâmetros
Corpo – instruções a executar

Exemplo

```
def print_disciplinas():  
    print( 'Sistemas Digitais')  
    print( 'Programação I')
```

Utilização (invocação)

Exemplo

```
>>> print_disciplinas()
```

```
Sistemas Digitais
```

```
Programação I
```

Notas

A definição cria a função

As instruções só são executadas quando a função é invocada

Uma função tem de ser definida antes de ser invocada

Fluxo de execução

A execução começa na 1ª instrução do programa

As instruções são executadas uma de cada vez, por ordem

A invocação de uma função provoca um desvio no fluxo de execução

Salta para o corpo da função

Executa as instruções lá existentes

Regressa, retomando o ponto onde tinha ficado

Nota

Ao analisar um programa devemos seguir o fluxo de execução

Resultado da função

Instrução return

Resultado indicado à direita da instrução

O programa continua com a instrução seguinte à invocação da função

Função void

Função que não devolve resultado

Argumentos e parâmetros

Argumento

Valor fornecido a uma função aquando da sua invocação

Parâmetro

Nome utilizado na função para referir o valor passado como argumento

Exemplo

```
def print_twice(xpto):  
    print(xpto)  
    print(xpto)  
>>> print_twice('Spam')  
Spam  
Spam  
>>> print_twice(17)  
17  
17
```


Visibilidade de variáveis e parâmetros

Variáveis e parâmetros são locais

São apenas visíveis na função onde foram definidos

Exemplo

```
>>> def soma_dois_valores( a, b ):  
    soma=a+b  
    return soma
```

```
>>> print( soma )
```

```
Traceback (most recent call last): print soma
```

```
NameError: name 'soma' is not defined
```

Visibilidade de variáveis e parâmetros

Podem existir funções diferentes com variáveis com o mesmo nome

... mas são entidades diferentes!

Uma variável local esconde outra com o mesmo nome

Dentro da função é usada a definição mais próxima do nome

Fora da função é usada a definição visível

Visibilidade de variáveis e parâmetros

```
>>>soma=8
```

```
>>>def soma_dois_valores(a,b):
```

```
    soma = a+b    # nova variável com o nome soma
```

```
    return soma
```

```
>>>resultado = soma_dois_valores(2,3)
```

```
>>>print( resultado )
```

```
5
```

```
>>>print( soma )
```

```
8
```

Expressões como argumentos

É feita a sua avaliação antes da execução do código da função

$f(g(1+2, h(3)), 4)$

Avaliar $g(1+2, h(3))$

Avaliar $1+2$

Avaliar $h(3)$

Executar a função h com argumentos: 3

Executar a função g com argumentos: 3 e resultado de $h(3)$

Executar a função f com argumentos: resultado de $g...$ e 4

Documentação

É recomendável adicionar documentação sobre a função

1ª linha do corpo da função

Delimitada por """

Opcional

Exemplo

```
def happy():  
    """Uma função que mostra um verso da canção de aniversário."""  
    print('Happy birthday to you!')
```

Pode ser consultada com a função help() (modo interativo)

```
>>> help(happy)
```

Porquê usar funções?

Porquê usar funções?

Torna o programa mais legível

Torna mais fácil fazer debugging

Permite analisar cada uma das partes em separado

Pode tornar o programa mais pequeno ao eliminar código repetido

Porquê usar funções?

Como programar?

Processo de desenvolvimento

Compreender o problema
Conceber o algoritmo
Implementar o algoritmo
Testar



Como aprender?

Estudar, estudar, ...
Praticar, praticar, ...
Cometer erros, cometer erros, ...
Aprender com os erros, ...