Fundamentos de Programação @ LEIC/LETI

Aula 05

Funções

Definição de funções. Aplicação de funções. Abstração procedimental. Erros. Módulos. Exemplos.

Alberto Abad, Tagus Park, IST, 2018

Funções

```
f(x, y) = x + y

f(3,5) = 8
```

- Igual que na Matemática, a utilização de funções em programação compreende a **definição** da função (nome, argumentos e algoritmo) e **aplicação de função** (execução do algoritmo sobre valores passados como argumentos).
- Exemplo funções Python já conhecidas: `print(...)`, `input(...)`, `eval(...)`

Definição de Funções (BNF)

```
<definição de função> ::=
    def <nome> (<parâmetros formais>): NEWLINE
    INDENT <corpo> DEDENT

<parâmetros formais> ::= <nada> | <nomes>

<nomes> ::= <nome> | <nome>, <nomes>

<nada> ::=

<corpo> ::= <definição de função>* <instruções em função>

<instruções em função> ::=
    <instrução em função> NEWLINE |
    <instrução em função> NEWLINE <instruções em função>

<instrução em função> ::= <instrução> | <expressão> | <instrução return>

<instrução return> ::= return | return <expressão>
```

Aplicação Funções (BNF)

```
<aplicação de função> ::= <nome>(<parâmetros concretos>)

<parâmetros concretos> ::= <nada> | <expressões>

<expressões> ::= <expressão> |
                 <expressão>, <expressões>
```

Definição e Aplicação de Funções, Exemplo 1:

```
def soma(a,b):
    return a + b
```

Exemplos:

- Aplicação antes e após definição.
- Aplicação: `soma(2)`, `soma(2,5`, `soma(7,5)`, `soma(3*2, 6+4)`
-
```
a, b = 2, 5
print("soma(a,b) =", soma(a,b))
print("soma(b,a) = ", soma(b,a))
print("a =",a,"b =",b)
```

Funções

Definição e Aplicação de Funções, Exemplo 1:

In [ ]:
```
def soma(a,b):
    return a + b

a = 5
b = 2
print(soma(1,1))
print(a,b)
```

Definição de Aplicação Funções, Exemplo 2:

```
def soma_progressao_aritmetica(n):
    iter = 1
    soma = 0
    while iter <= n:
        soma = soma + iter
        iter = iter + 1
    return soma

soma = 4 + 6
print(soma_progressao_aritmetica(100), soma)
```

- Que acontece com a variável *soma*!?!?

Definição e Aplicação de Funções, Exemplo 2:

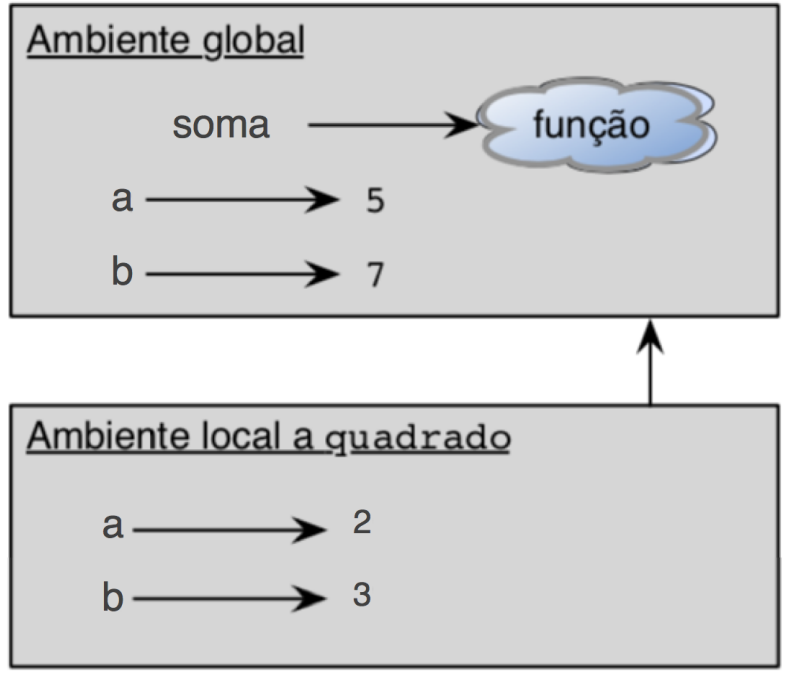
In [ ]:
```
def soma(n):
    if n < 1:
        return 0
    return n*(n+1)//2

soma(5)
soma(7)
```

Ambientes e quadros (*frames*)

- Ambientes: Global vs. Local
- Passos seguidos pelo Python quando uma função é invocada:
  - Os parâmetros concretos são avaliados (ordem arbitrária)
  - Os parâmetros formais da função são associados com os valores concretos no ambiente local (em ordem)
  - O corpo da função é executado no ambiente local (os ambientes locais existem só até a função terminar) e o valor de *return* é retornado ao ambiente global

Ambientes e quadros (*frames*)



Abstração procedimental

- As funções permitem aos programadores pensar no **que** (faz a função) e não no **como** (a função é implementada).

Exemplo

```
def soma(n):
    iter = 1
    soma = 0
    while iter <= n:
        soma = soma + iter
        iter = iter + 1
    return soma
```

```
def soma(n):
    if n < 1:
        return 0
    else:
        return n*(n+1)//2
```

Erros/Excepções

- Nas semanas anteriores falamos dos tipos de erros: sintaxe, semântica e *runtime*
- As funções podem *lançar* erros quando os argumentos utilizados são de tipo inválido e/ou estão fora do dominio.
  - As excepções interrompen o fluxo de execução, o que não acontece se fizermos um simples *print*
- Para isso podemos utilizar a instrução *raise* que gera um erro de execução, em BNF:

```
<instrução raise> ::= raise <nome>(<mensagem>)

<mensagem> ::= <cadeia de caracteres>
```

Erros/Excepções

Nome	Situação correspondente ao erro
`AttributeError`	Referência a um atributo não existente num objeto.
`ImportError`	Importação de uma biblioteca não existente.
`IndexError`	Erro gerado pela referência a um índice fora da gama de um tuplo ou de uma lista.
`KeyError`	Referência a uma chave inexistente num dicionário.
`NameError`	Referência a um nome que não existe.
`SyntaxError`	Erro gerado quando uma das funções `eval` ou `input` encontram uma expressão com a sintaxe incorreta.
`ValueError`	Erro gerado quando uma função recebe um argumento de tipo correto mas cujo valor não é apropriado.
`ZeroDivisionError`	Erro gerado pela divisão por zero.

Tabela 3.3: Alguns dos identificadores de erros em Python.

- Python (como outras linguagens) fornecem um *protocol* para tratar das excepções (*try/except*) que veremos nas próximas semanas

Erros/Excepções, Exemplo:

In [16]:
```
def inverte(n):
    if not (type(n) == int):
        raise ValueError("erro nao numero")
    elif n == 0:
        raise ValueError("erro igual a 0")
    return 1/n


i(3)
```

Out[16]:
```
0.3333333333333333
```

Módulos: Importar

- Não é preciso reiventar a roda, Python fornece um grande número de bibliotecas (*libraries*) ou módulos com funções que podemos importar:
- Lista de módulos disponíveis por omissão: https://docs.python.org/3/py-modindex.html (https://docs.python.org/3/py-modindex.html)

```
<instrução import> ::=
    import <módulo> {as <nome>} NEWLINE |
    from <módulo> import <nomes a importar> NEWLINE

<módulo> ::= <nome>

<nomes a importar> ::= * | <nomes>

<nomes> ::= <nome> | <nome>, <nomes>
```

Módulos: Aceder funções dum módulo

- Necessário no caso de *import* (sem *from*):

```
<composed name> ::= <simple name>.<simple name>
```

Exemplos:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(math.pi/2)
1.0
```

```
>>> from math import pi, sin
>>> pi
3.141592653589793
>>> sin(pi/2)
1.0
```

Módulos: Construir módulos

- Colocar funções num ficheiro *.py* (ex: soma.py)
- Importar utilizando o nome do ficheiro/módulo (sem extensão):

```
>>> import soma
>>> soma.soma(100)
5050
```

Funções e parâmetros em Python ++ (opcional)

- Python permite maior flexibilidade na definição e passagem dos parâmetros duma função:
  - **Default parameters**
  - **Keyword arguments**
  - Número variável de parâmetros posicionais e keyword (não neste curso)

In [ ]:
```
def dividir(num, den = 1):
    return num/den
print("Ex1:", dividir(10,2))
print("Ex2:", dividir(10))
```

Visualização e execução de programas

- http://pythontutor.com/visualize.html#mode=edit (http://pythontutor.com/visualize.html#mode=edit)
- IDEs como o PyCharm e WingIDE
- Demo?

A treinar!!!!

Funções

Exemplo 1, Tabela conversão temperaturas

In [ ]:

```
def far_para_cent(F):
    return round(5 * (F - 32) / 9)

min = eval(input('Qual a temperatura minima?\n? '))
max = eval(input('Qual a temperatura maxima?\n? '))

while min <= max:
    print(min, '=', far_para_cent(min))
    min = min + 1
```

Exemplo 2, Potência de dois números inteiros

In [ ]:
```
# Power of two numbers inteiros
def potencia(x, k):
    if k < 0:
        raise ValueError('potencia: exponente k negativo')
    elif type(k) != int:
        raise ValueError('potencia: exponente k nao inteiro')
    elif type(x) != int:
        raise ValueError('potencia: exponente x nao é um inteiro')

    product = 1

    while k > 0:
        product = product*x
        k = k - 1

    return product

print(potencia(3, 4))
```

Exemplo 3, Factorial

In [ ]:
```
# factorial
def factorial(n):
    if type(n) != int:
        raise ValueError("Não inteiro!")
    elif n < 0:
        raise ValueError("Negativo!")

    f, i = 1, 1
    while i <= n:
        f = f * i
        i = i + 1
    return f

    # Alternativa
    # f = 1
    # while n > 0:
    #     f = f * n
    #     n = n - 1
    # return f

x = eval(input("Inteiro: "))
f = factorial(x)
print(f)
```

Exemplo 4, Máximo divisor comum (Algoritmo de Euclides)

1. O máximo divisor comum entre um número e zero é o próprio número: `mdc(m,0) = m`
2. Quando dividimos um número *m* por um menor *n*, o máximo divisor comum entre o resto da divisão e o divisor é o mesmo que o máximo divisor comum entre o dividendo e o divisor: `mdc(m, n) = mdc(n, m%n)`

- Exemplo algorimo para mdc(24, 16):

m	n	m % n
24	16	8
16	8	0
8	0	8

Exemplo 4, Máximo divisor comum (Algoritmo de Euclides)

In [ ]:
```
# Máximo divisor comum (mdc)
# Euclidian algorithm

def mdc(m, n):

    if n < 0 or m < 0:
        raise ValueError('euclides: argumentos negativos!')

    while n > 0:
        m, n = n, m%n

    return m

x = eval(input("Da-me valor x:"))
y = eval(input("Da-me valor y:"))
print(mdc(x, y))
```

Exemplo 5, Raiz quadrada (Algoritmo da Babilónia)

- Em cada iteração, partindo do valor aproximado, p_i, para a raiz quadrada de x, podemos calcular uma aproximação ao melhor, p_{i+1}, através da seguinte fórmula:

$$p_{i+1} = \frac{p_i + \frac{x}{p_i}}{2}.$$

- Exemplo algoritmo para $\sqrt{2}$

Número da tentativa	Aproximação para $\sqrt{2}$	Nova aproximação
0	1	$\frac{1+\frac{2}{1}}{2} = 1.5$
1	1.5	$\frac{1.5+\frac{2}{1.5}}{2} = 1.4167$
2	1.4167	$\frac{1.4167+\frac{2}{1.4167}}{2} = 1.4142$
3	1.4142	...

Exemplo 5, Raiz quadrada (Algoritmo da Babilónia)

```
def calcula_raiz(x, palpite):
    while not bom_palpite(x, palpite):
        palpite = novo_palpite(x, palpite)
    return palpite

def raiz(x):
    if x < 0:
        raise ValueError("raiz definida só para números positivos")
    return calcula_raiz(x, 1)
```

- **Exercício**: Definir as funções *bom_palpite* e *novo_palpite*

Exemplo 6, Séries de Taylor

- Definição:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3$$

- Exemplos dalgumas aproximações:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

Exemplo 6, Séries de Taylor

```
def proximo_termo(x, n):
    pass #completar (diferente dependendo da função a aproximar)

def funcao_aproximada(x, delta):
    n = 0
    termo = proximo_termo(x, n)
    resultado = termo
    while termo > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo
    return resultado
```

- **Exercício:** Definir a série de Taylor para as funçoes *e(x)*, *sin(x)* e *cos(x)*
- **Exercício:** Alterar para que o cômputo de termo seja função do anterior termo, `termo = proximo_termo(x, n, termo)`

Exemplo 6, Séries de Taylor: Exponencial

In [ ]:
```
def exp_aproximada(x, delta):
    def proximo_termo(x, n):
        return x**n/factorial(n)

    n = 0
    termo = proximo_termo(x, n)
    resultado = termo

    while termo > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo

    return resultado

print("Aprox",exp_aproximada(3,0.001))
from math import exp
print("Exacto",exp(3))
```