Ciências ULisboa Faculdade de Ciências da Universidade de Lisboa
DEPARTAMENTO·DE·INFORMÁTICA

Folha 5 - Funções

Python v 3.*

Exercícios

1. Considere a seguinte função:

```
def imprimeDivisaoInteira(x, y):
     if y == 0:
          print("Divisão por zero”)
     else:
          print(x//y)
```

a) O que faz esta função?

b) Qual o resultado da seguinte sequência de comandos?

```
imprimeDivisaoInteira(4, 2)
imprimeDivisaoInteira(2, 4)
imprimeDivisaoInteira(3, 0)
help(imprimeDivisaoInteira)
imprimeDivisaoInteira()
```

c) Altere a definição da função de modo a adoptar a abordagem da programação por contratos (recorrendo a uma docstring) em vez da abordagem da programação defensiva.

2. Considere a seguinte função:

```
def divisaoInteira(x, y):
     """
     Divide dois inteiros dados

     Requires: x e y sejam int e y!=0
     Ensures: um int correspondente à divisão inteira
     de x por y
     """

     return x//y
```

a) Qual a maior diferença entre as funções **imprimeDivisaoInteira** (exercício anterior, considerando a versão alterada na alínea **1c**) e **divisaoInteira**?

b) Qual o resultado da seguinte sequência de comandos?

```
imprimeDivisaoInteira(4, 2)
a = divisaoInteira(6, 2)
print(a)
b = a + divisaoInteira(10,5)
print(b)
c = divisaoInteira(4, divisaoInteira(1,2))
print(c)
print(divisaoInteira(4+4, 2+2))
```

3. Considere a seguinte função:

```
def potencia(a, b):
     return a**b
```

a) O que faz a seguinte sequência de comandos?

```
a = 2
b = 3
potencia(b, a)
potencia(a, b)
print(potencia(b, a))
print(potencia(a, b))
print(potencia(2,0))
print(potencia(2))
```

b) Escreva uma nova função **potenciaP** que receba apenas um número inteiro *a* e retorne a^a.

c) Escreva uma nova versão da função **potencia** que possa ser chamada com dois ou com um argumento e, consoante o caso, execute a^b ou a^a.

4. Considere a seguinte função:

```
def calculaOrdenado(base, irs=0.30, ss=0.10, outros=0):
    """
    Calcula o ordenado líquido (após descontos)

     Calcula o ordenado líquido dados o valor do ordenado bruto
     (base) e duas taxas de descontos, a do irs (irs), a da segurança
     social (ss), e ainda o montante de outros descontos (outros)
     Requires: base é float, base > 0, 0<=irs<=1, 0<=ss<=1
     Ensures: um float com o valor do ordenado líquido
     """

     descontos = base * irs + base * ss + outros
```

```
    return base – descontos
```

a) O que são argumentos passado por palavra-chave (*keyword*) e o que são valores de argumentos por omissão (*default*)?

b) Qual o resultado de cada um dos seguintes comandos?

```
print(calculaOrdenado(1000))
print(calculaOrdenado(1000, 0, 0, 200))
print(calculaOrdenado(1000, outros = 200))
print(calculaOrdenado(1000, 0.20, outros = 100))
print(calculaOrdenado(1000, ss=0.20, outros = 100))
print(calculaOrdenado(1000, irs=0.20, 100))
```

c) Altere a função **calculaOrdenado** de forma a poder receber um argumento Booleano que indica se os outros descontos devem ser efectuados ou não. Por omissão, os outros descontos são efectuados.

d) Apresente dois exemplos de utilização da função **calculaOrdenado** em que num exemplo os outros descontos são aplicados, e noutro não.

5. Relembre a equação da recta na forma $y = mx + b$ onde m é o declive da recta. Considere a função seguinte que calcula *y* usando valores fixos de *x, m e b.* Altere a função de forma a generalizá-la para quaisquer valores dados de *m, x e b.*

```
def y():
    m = 5
    x = 9
    b = -3
    return m * x + b
```

6. Considere o seguinte programa:

```
a = 4
def printFuncao():
    a = 17
    print("Dentro da função: ", a)

printFuncao()
print("Fora da função: ", a)
```

a) Qual é o resultado de executar este programa?

b) O que é uma variável global? E uma variável local?

7. Considere o seguinte programa:

```
a = 10
b = 15
c = 25

def f(a):
    print("Dentro: a=", a)
    b = 100 + a
    d = 2 * a
    print("Dentro: b=", b)
    print("Dentro: c=", c)
    print("Dentro: d=", d)
    return b + 10

c = f(b)

print("Fora: a = ", a)
print("Fora: b = ", b)
print("Fora: c = ", c)
print("Fora: d = ", d)
```

Qual o resultado da execução do programa?

8. Considere o seguinte código:

```
def somaDivisores(num):
    """
    Soma de divisores de um numero dado

    Requires: num seja int e num > 0
    Ensures: um int correspondente à soma dos divisores
    de num que sejam maiores que 1 e menores que num
    """
```

a) Como cliente da função **somaDivisores**, o que deve satisfazer para a função cumprir o contrato?

b) E o que obtém da função se a chamar satisfazendo a sua pré-condição?

9. Crie um programa que pergunte sucessivamente ao utilizador um número inteiro positivo e imprima a soma dos seus divisores. A execução do programa deve terminar quando o utilizador introduzir um número negativo.
Nota: assuma que a função **somaDivisores**, apresentada no exercício anterior, se encontra definida.

10. Considere o seguinte programa:

```
DIA_ATUAL = 2
MES_ATUAL = 11
ANO_ATUAL = 2015

print("Dados do Pai")
anoPai = int(input("Introduza o ano de nascimento: "))
mesPai = int(input("Introduza o mes de nascimento: "))
diaPai = int(input("Introduza o dia de nascimento: "))

print("Dados da Mãe")
anoMae = int(input("Introduza o ano de nascimento: "))
mesMae = int(input("Introduza o mes de nascimento: "))
diaMae = int(input("Introduza o dia de nascimento: "))

if mesPai > MES_ATUAL or \
    (mesPai == MES_ATUAL and diaPai > DIA_ATUAL):
    print("Pai tem", ANO_ATUAL - anoPai - 1, "ano(s)")
else:
    print("Pai tem", ANO_ATUAL - anoPai, "ano(s)")

if mesMae > MES_ATUAL or \
    (mesMae == MES_ATUAL and diaMae > DIA_ATUAL):
    print("Mãe tem", ANO_ATUAL - anoMae - 1, "ano(s)")
else:
    print("Mãe tem", ANO_ATUAL - anoMae, "ano(s)")
```

Recorrendo a funções, simplifique o programa apresentado eliminando a repetição de código.

Problemas

1. Escreva uma função com nome **succ** que receba um número inteiro e devolva o inteiro seguinte, i.e., o sucessor do número dado. Escreva o contrato da função usando uma docstring: não esquecer de incluir a descrição da função, a descrição e tipo do parâmetro e a descrição e o tipo do valor que a função devolve. Teste a função incluindo no seu programa o comando de impressão do resultado da chamada à função **succ(-1).** Experimente fazer **help(succ)** no modo interativo do interpretador Python.

2. Escreva uma função que receba dois números inteiros e devolva o maior deles. Inclua o contrato da função. Teste a função escrevendo um programa que receba dois números inteiros do utilizador e imprima o resultado da chamada à função desenvolvida. Como teria de fazer para determinar o menor de dois números com uma segunda função que tirasse partido de chamar a primeira?

3. Escreva uma função que devolva o maior de três números inteiros (deve incluir o contrato na escrita). Recorra à função implementada no exercício anterior. Teste a função escrevendo um programa que receba três números inteiros do utilizador e imprima o resultado da chamada à função desenvolvida.

4. Escreva uma função **unidades** (escrever o contrato faz parte da escrita da função) que devolva o dígito na casa das unidades de um número inteiro. Por exemplo **unidades(43)** deve devolver 3. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado de chamada à função desenvolvida.

5. Escreva uma função (o contrato não pode ser esquecido) que elimine a casa das unidades de um número inteiro. Por exemplo, **retira(537**) devolve 53. Se o argumento só tiver algarismo das unidades, a função deve devolver 0. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado de chamada à função desenvolvida.

6. Escreva uma função que acrescente um 0 no final de um número inteiro. Por exemplo, **aumenta(73)** devolve 730 (se esquecer o contrato a definição da função está incompleta). **aumenta(0)** deve devolver 0. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado da chamada à função desenvolvida.

7. Escreva uma função que verifique se um dado número inteiro é uma capicua usando as funções desenvolvidas nos três exercícios anteriores. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado da chamada à função desenvolvida.

8. Escreva uma função que receba um número inteiro e devolva a soma dos divisores próprios desse número:

```
def somaDivisores(num):
    """Soma dos divisores próprios de um número dado.

    Requires: num seja int e num > 0
    Ensures: um int correspondente à soma dos divisores
    de num que sejam maiores que 1 e menores que num
    """
```

Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado da chamada à função desenvolvida.

9. Escreva uma função que verifique se um dado número inteiro é perfeito. Relembre que um número é perfeito se é igual à soma dos seus divisores próprios mais 1. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado da chamada à função desenvolvida.

10. Escreva uma função que verifique se um dado número dado é primo. Relembre que um número é primo se é maior do que 1 e não tem divisores próprios. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado da chamada à função desenvolvida.

11. Usando a função do exercício anterior, escreva um programa que receba um número inteiro *n* maior do que 2 e escreva no ecrã quantos números primos existem entre 2 e *n* (inclusive). Por exemplo, existem 1000 números primos entre 2 e 7919. Teste a função escrevendo um programa que receba um número inteiro do utilizador e imprima o resultado de chamada à função desenvolvida. Explique, nesta situação, em que difere a programação defensiva da programação por contratos.