

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

1º Ano/1º Semestre

Unidade Curricular: Introdução à Programação

Docente: Frederica Gonçalves

1

FUNÇÕES

O processo de utilização de funções corresponde a dois processos distintos:

Definição de função, que, é feita fornecendo o nome da função, os argumentos da função e o processo de cálculo para os valores da função (*processo que é descrito por um algoritmo*).

Aplicação da função, a um valor ou valores, do (s) seu (s) argumento (s). Esta aplicação corresponde à execução do algoritmo associado à função para valores particulares, designada por *chamada à função*.

Para definir funções em Python, é necessário indicar o **nome da função**, os seus **argumentos** (designados por *parâmetros formais*) e o **processo de cálculo (algoritmo)** dos valores da função (designado por *corpo da função*). Exemplo:

```
def quadrado (x):  
    return x * x
```

quadrado – nome da função

(x) – argumento da função

return x * x – corresponde ao corpo da função

Nota:

Um modo interessante de seguir o funcionamento de um programa em Python corresponde utilizar a aplicação Python Tutor, disponível em <http://pythontutor.com/visualize.html>

Exercícios:

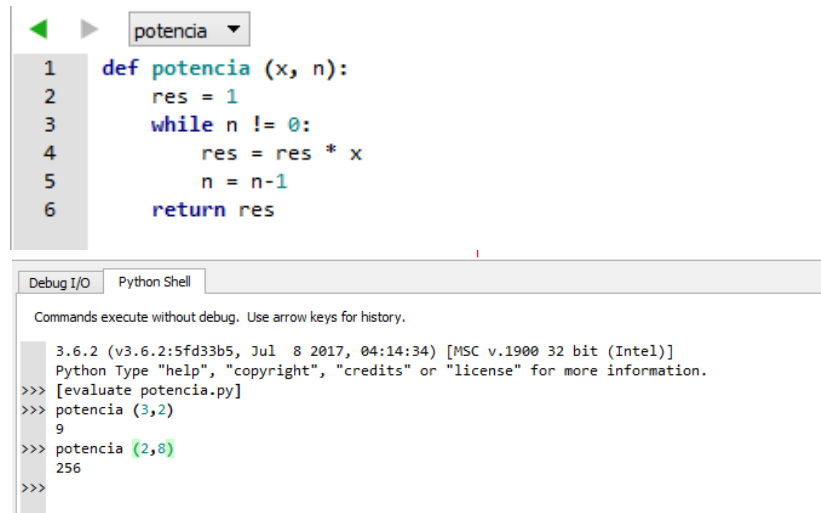
Cofinanciado por:

1. Crie um programa que cria uma tabela de conversão de graus Fahrenheit para graus centígrados, entre dois valores fornecidos pelo utilizador.

```
1 def far_para_cent (F):
2     return round (5*( F-32)/ 9 )
3 min = eval (input ('Qual a temperatura mínima?\n?'))
4 max = eval (input ( 'Qual a temperatura máxima ?\n?'))
5 while min <= max:
6     print (min, 'F=', far_para_cent (min), 'C')
7     min = min +1
```

2

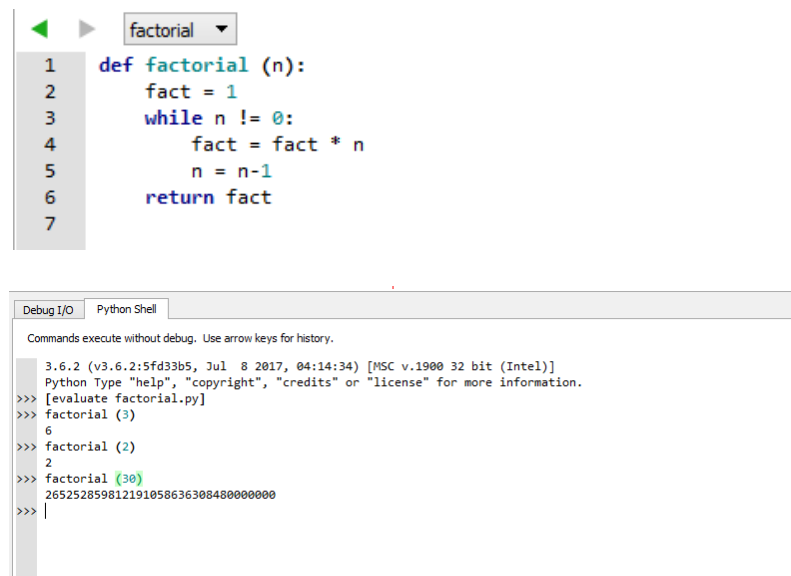
2. Crie um programa que calcule a potência de um número.



```
potencia ▼
1 def potencia (x, n):
2     res = 1
3     while n != 0:
4         res = res * x
5         n = n-1
6     return res

Debug I/O Python Shell
Commands execute without debug. Use arrow keys for history.
3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
Python Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate potencia.py]
>>> potencia (3,2)
9
>>> potencia (2,8)
256
>>>
```

3. Crie um programa para calcular o factorial de um inteiro.



```
factorial ▼
1 def factorial (n):
2     fact = 1
3     while n != 0:
4         fact = fact * n
5         n = n-1
6     return fact
7

Debug I/O Python Shell
Commands execute without debug. Use arrow keys for history.
3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
Python Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate factorial.py]
>>> factorial (3)
6
>>> factorial (2)
2
>>> factorial (30)
265252859812191058636308480000000
>>>
```

MÓDULOS

Cofinanciado por:



Um módulo (também conhecido por biblioteca) é uma coleção de funções agrupadas num único ficheiro. As funções existentes no módulo estão relacionadas entre si.

Exemplo: Muitas das funções matemáticas (raiz quadrada, seno, etc.) estão definidas no módulo `math`.

Para utilizar um módulo é necessário *importar*.

As funções do módulo são referenciadas através de uma variação de nomes chamada **nome composto**.

Um nome composto, corresponde à especificação do nome do módulo, seguido por um ponto, seguido pelo nome da função.

3

```
>>> import math
>>> math.pi
3.141592653589793
```

Através da instrução de importação `from math import pi, sin` (por exemplo) permite-nos importar quais as funções ou nomes a importar do módulo.

Se utilizarmos `from math import *`, então todos os nomes do módulo são importados para o programa.

Atenção: Pode acontecer que dois módulos diferentes utilizem o mesmo nome para referirem funções diferentes.

Cofinanciado por:

