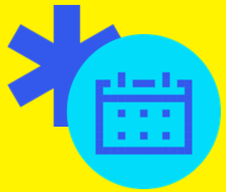
blue

Funções e Módulos – Python

< thefutureisblue.me />



Agenda

O que vamos aprender?

- Funções
- Módulos

O que é abstração?

* Técnica de programação que nos permite pensar num problema em diversos níveis.
* Quando pensamos num problema macroscopicamente, não estamos preocupado com minúcias.
* Dividir para conquistar:
  - Um problema é dividido em diversos subproblemas.
  - As soluções dos subproblemas são combinadas numa solução do problema maior.

O que é programação estruturada?

* Disciplina incorpora o princípio “dividir para conquistar”.
* Programas são divididos em subprogramas.
 - Cada subprograma é invocado por meio de um identificador e de uma lista de entradas.
- Permite especificar como um problema pode ser resolvido, em geral.
 - O subprograma pode ser invocado para resolver problemas de mesma natureza, mas com valores diferentes.
* Os resultados computados por um subprograma podem ser combinados com os de outros subprogramas.

Definindo funções!

* Na linguagem Python, subprogramas têm o nome de **funções**:

```
def <nome_função> (<definição dos parâmetros>) :
        <Bloco de comandos da função>
```

-> <nome_função> é o nome da função.
-> definição de paramêtros são especificações de argumentos da função (uma função pode ter 0, 1 ou mais argumentos).
-> Bloco de comandos contêm as instruções a serem executadas.

Resultados de funções!

- Uma função tipicamente computa um ou mais valores.
- Para indicar o valor a ser devolvido como o resultado da função, usa-se o comando **return**, que tem o formato **return** <expressão> (em que a <expressão> é opcional e designa o valor a ser retornado).
- Ao encontrar o comando **return**, a função termina imediatamente e o controle do programa volta ao ponto em que a função foi chamada.
- Se uma função chega a seu fim, sem nenhum valor de retorno ter sido especificado, o valor de retorno é **None**.

Exemplo de funções!

*
```
def f1():
    return
f1()
# Saída: None
```

*
```
def f1():
    print("Hello World")
f1()
# Saída: Hello World
```

*
```
def f(nome):
    return "Olá, " + nome + "!"
f("Marcio")
# Saída: Olá, Marcio!
```

Exemplo de parâmetros e argumentos!

* Parâmetros são as variáveis que podem ser incluídas nos parênteses das funções. Quando a função é chamada são passados valores para essas variáveis. Esses valores são chamados argumentos. O corpo da função pode utilizar essas variáveis, cujos valores podem modificar o comportamento da função.

```
def maior(b, e):
    if b > e:
        print(b)
    else:
        print(e)
maior(6, 9)
# Saída: 9
```

Exemplo de escopo das variáveis!

* Toda variável utilizada dentro de uma função tem escopo local, isto é, ela não será acessível por outras funções ou pelo programa principal. Se houver variável com o mesmo nome fora da função, será uma outra variável, completamente independentes entre si.

```
def soma(x, y):
    total = x + y
    print("Total soma = ", total)

# programa principal
total = 10
soma(3, 5)
print("Total principal = ", total)

# Saída: Total soma = 8
Total principal = 10
```

Exemplo de escopo das variáveis!

* Para uma variável ser compartilhada entre diversas funções e o programa principal, ela deve ser definida como variável global. Para isto, utiliza-se a instrução global para declarar a variável em todas as funções para as quais ela deva estar acessível. O mesmo vale para o programa principal.

```
def soma(x, y):
        global total
        total = x + y
 print("Total soma = ", total)

# programa principal
global total
total = 8
soma(3, 5)
print("Total principal = ", total)
# Saída: Total soma = 8
Total principal = 8
```

Exemplo de retorno de valores!

- O comando return é usado para retornar um valor de uma função e encerrá-la. Caso não seja declarado um valor de retorno, a função retorna o valor None (que significa nada, sem valor).

```
def soma(x,y):
    total = x+y
    return total

# programa principal
S = soma(3, 5)
print("soma = ", s)
# Saída: soma = 8
```

Observações:

a) O valor da variável total, calculado na função soma, retornou da função e foi atribuído à variável s.

b) O comando após o return foi ignorado.

Exemplo de valor padrão!

* É possível definir um valor padrão para os parâmetros da função. Neste caso, quando o valor é omitido na chamada da função, a variável assume o valor padrão.

```
def calcula_juros(valor, taxa=10):
    juros = valor * taxa / 100
    return juros

calcula_juros(500)
# Saída: 50.0
```

O que são módulos?

São programas feitos para serem reaproveitados em outros programas.
Eles contêm funções, variáveis, classes e objetos que provêm alguma funcionalidade comum (por exemplo, o módulo math contém funções matemáticas comosin, exp, etc).
Toda a biblioteca do Python é dividida em módulos e pacotes.
Alguns dos mais comuns são: sys, os, time, random, re, shelve.

Escrevendo um módulo!

- Qualquer programa que você escreva e salve num arquivo pode ser importado como um módulo.
- Por exemplo, se você salva um programa com o **nome mod1.py**, ele pode ser importado usando o comando:

```
import mod1
```

- Entretanto a “importação” só ocorre uma vez.
- Python assume que variáveis e funções não são mudados e que o código do módulo serve meramente para inicializar esses elementos.

Escrevendo um módulo!

- Após a importação, o módulo é compilado, gerando um arquivo **.pyc** correspondente.
- No exemplo, um arquivo **mod1.pyc** será criado.
- Python só recompila um programa se o arquivo .py for mais recente que o arquivo .pyc

Escrevendo um módulo!

* Conteúdo do arquivo mod1.py:
*
```
def f():
    print"Hello Module!"
    f()
```

* Execução pelo console Python:

```
import mod1.py
Hello Module!
```

* Se observarmos do diretório em que foi executado, teremos agora dois arquivos o mod1.py e o mod1.pyc.

blue

Botando para rodar!

Vamos praticar todos operadores conceituados!

the future is blue

Por hoje é só!
Obrigado! =)

Até a próxima aula.