**CEUB - FATECS**

**Lógica de Programação - Prof. Barbosa**

**Python - Função (*function*) - def**

**A função em Python é definida com a palavra-chave def.**

**- Significado da palavra-chave def:**

**a. É uma palavra reservada da linguagem Python que significa define (definir).**

**b. Indica o início da definição de uma função.**

**c. Permite criar código reutilizável e organizado.**

**O def é usada para definir uma função, ou seja, criar um bloco de código nomeado que pode ser reutilizado em diferentes partes do programa.**

**- Nome da função:**

**a. Deve ser um nome intuitivo (significativo);**

**b. O padrão (convenção) é usar todas as letras minúsculas e o underline (sublinha) quando tiver mais de uma palavra.**

**- Sintaxe:**

**def nome\_funcao( [par1, par2, ... , parn] ): # Os parâmetros são opcionais**

**# Bloco de código da função # Indentação obrigatória**

**[return valor] # O return é opcional**

**- Obs.:**

**Uma função pode ou não receber parâmetros.**

**Uma função pode ou não retornar um valor.**

**Na sintaxe do comandos de TI, os colchetes [ ] indicam a parte opcional do comando.**

**- Chamar (usar) a função criada, dentro do main (programa):**

**# O if abaixo indica o início da execução do programa principal (main)**

**if \_\_name\_\_ == '\_\_main\_\_': # Atalho: mai + tecla <tab>**

**. . . # Indentação obrigatória.**

**nome\_funcao( [arg1, arg2, . . . , argn] ) # Chama a função passando os argumentos.**

**. . .**

**Obs.: um programa pode ter várias funções (defs), mas apenas uma função principal (main).**

**A convenção acima ( \_\_ ) utilizada em Python e é chamada de "dunder", vem da expressão em inglês "double-underscore" - duplo sublinhado em português.**

**- Notação:**

**def - palavra-chave usada no início da função, declara que tem uma função**

**nome\_funcao - nome da função definido pelo desenvolvedor, use nome significativo**

**[arg1, arg2, ... , argn] - arg (argumento) - os dados que são enviamos para a função, opcional**

**[par1, par2, ... , parn] - parâmetros (valores que a função recebe), opcional**

**script - código Python dentro da função**

**return - palavra-chave usada no fim da função, retorna o valor calculado, opcional**

**valor - valor retornado pela função**

**[ ] - indica que essa parte da sintaxe é opcional**

**1- Sintaxe de um programa com a função main (programa) e uma função def sem return:**

**def nome\_funcao( [par1, par2, ... , parn] ): # Os parâmetros são opcionais**

**# Bloco de código da função # Indentação obrigatória**

**...**

**print(“Msg”, valor)**

**[return valor] # Sem return**

**- Chamar (usar) a função criada, dentro do main (programa):**

**if \_\_name\_\_ == '\_\_main\_\_': # Atalho: mai + tecla <tab>**

**. . .**

**nome\_funcao([arg1, arg2, ... , argn])** **# Chama a função a primeira vez**

**. . .**

**nome\_funcao([arg1, arg2, ... , argn]) # Chama a função a segunda vez**

**. . .**

**Obs.: uma função pode ser chamada mais de uma vez no main.**

**- Observações:**

**Uma função nada mais é do que uma sub-rotina usada em um programa.**

**-2 Sintaxe de um programa com a função main (programa) e uma função def com return:**

**def nome\_funcao( [par1, par2, ... , parn] ): # Os parâmetros são opcionais**

**# Bloco de código da função # Indentação obrigatória**

**return valor # Com return**

**- Chamar (usar) a função criada, dentro do main (programa):**

**if \_\_name\_\_ == '\_\_main\_\_': # Atalho: mai + tecla <tab>**

**. . .**

**v\_ retorno1 = nome\_funcao([arg1, arg2, ... , argn]) # Chama a função a primeira vez**

**print(“Msg:”, v\_retorno1)**

**. . .**

**v\_ retorno2 = nome\_funcao([arg1, arg2, ... , argn]) # Chama a função a segunda vez**

**print(“Msg:”, v\_retorno2)**

**. . .**

**- Declarar função vazia:**

**Precisa escrever algo no escopo da função, caso contrário o PyCharm aponta um erro.**

**def funcao\_vazia(): # A função não recebe nada.**

**pass # ...**

**- Chamar (usar) a função criada, dentro do main (programa):**

**if \_\_name\_\_ == '\_\_main\_\_': # Atalho: main + tecla <tab>**

**. . .**

**funcao\_vazia() # Chama a função sem passar nada.**

**. . .**

**- Exemplo:**

**print() é uma função nativa do Python, ou seja, a ideia de criar uma função é a sua reutilização em vários momentos do programa.**

**Prof. Barbosa**