



INTRODUÇÃO À PROGRAMAÇÃO COM A LINGUAGEM PYTHON

PARTE II: PYTHON FUNDAMENTAL

FABRÍCIO G. M. DE CARVALHO

Atualização: Junho 2019

1

ESTRUTURA DO CURSO

PARTE II: PYTHON FUNDAMENTAL

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

- a) ESTRUTURA BÁSICA DE UM PROGRAMA
- b) SAÍDA DE DADOS
- c) VARIÁVEIS E TIPOS
- d) ENTRADA DE DADOS
- e) ESTRUTURAS DE CONTROLE
- f) ESTRUTURAS DE REPETIÇÃO
- g) FUNÇÕES

2

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

a) ESTRUTURA BÁSICA DE UM PROGRAMA

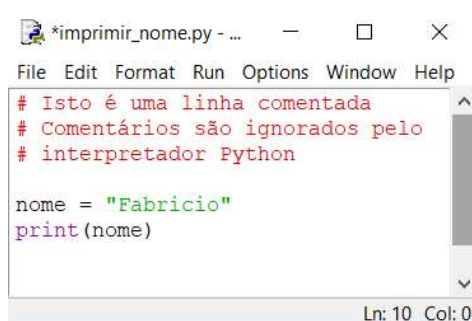
- ✓ Um programa feito em Python, consiste de um ou mais arquivos com extensão .py contendo o código ser executado.
- ✓ Os arquivos podem estar localizados na mesma pasta ou em pastas distintas (pacotes).
- ✓ Cada arquivo deve conter uma série de comandos que devem ser executados pelo interpretador.

3

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

a) ESTRUTURA BÁSICA DE UM PROGRAMA (cont.)

Exemplo 1: Utilizando o Idle, criar um arquivo chamado imprimir_nome.py com o conteúdo mostrado abaixo. Executar via Idle (Run) e prompt de comando.



```

# Isto é uma linha comentada
# Comentários são ignorados pelo
# interpretador Python

nome = "Fabricio"
print(nome)

```

4

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

b) SAÍDA DE DADOS

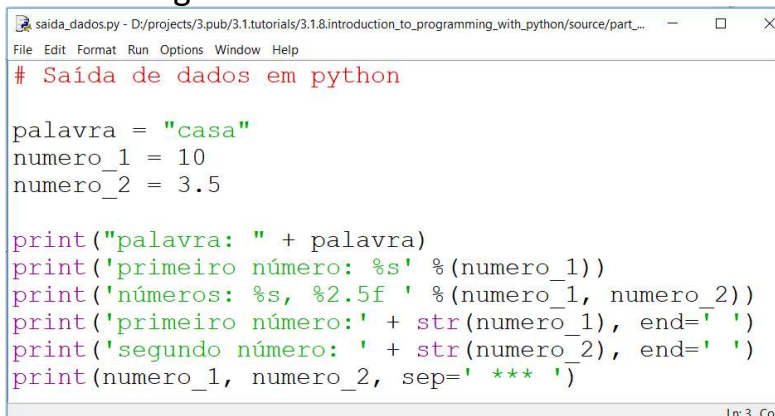
- ✓ A maneira mais simples de se dispor os dados a partir da aplicação é através da utilização da função `print()`.
- ✓ Essa função possui comportamento que depende do tipo do dado informado e também da forma como os argumentos são passados.

5

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

b) SAÍDA DE DADOS

Exemplo 2: Criar um arquivo `saída_dados.py` com o conteúdo a seguir e executá-lo.



```

saída_dados.py - D:/projects/3.pub/3.1.tutorials/3.1.8.introduction_to_programming_with_python/source/part_...
File Edit Format Run Options Window Help
# Saída de dados em python

palavra = "casa"
numero_1 = 10
numero_2 = 3.5

print("palavra: " + palavra)
print('primeiro número: %s' %(numero_1))
print('números: %s, %2.5f ' %(numero_1, numero_2))
print('primeiro número: ' + str(numero_1), end=' ')
print('segundo número: ' + str(numero_2), end=' ')
print(numero_1, numero_2, sep=' *** ')
Ln: 3 Col

```

6

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

c) VARIÁVEIS E TIPOS

- ✔ Em Python, de um modo geral, as variáveis não são verificadas estaticamente e possuem tipos que podem ser dinamicamente definidos, ou seja, uma variável que armazena um número pode passar a armazenar uma cadeia de caracteres, um objeto mais complexo, etc.
- ✔ Uma variável é criada definindo-se seu nome e, opcionalmente, atribuindo-se a ela um valor.

7

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

c) VARIÁVEIS E TIPOS

Exemplo 3: Criar e executar o programa a seguir.

```
# Variáveis e tipo

#Cadeia de caracteres (String)
variavel = 'Fabrício'
print(variavel)

#inteiro
variavel = 1
print(variavel)

#Dicionário
variavel = {'nome': 'Fabricio', 'idade': 40}
print(variavel)
print(variavel['nome'], variavel['idade'], sep=' // ')
```

8

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

c) VARIÁVEIS E TIPOS

- ✓ Além de números, cadeias de caracteres e dicionários, Python suporta listas, objetos, etc., como valores que podem ser atribuídos a uma variável.

Exemplo 4: Criar e executar o programa a seguir.

```
# Exemplificando listas
frutas = []
frutas.append("abacate")
frutas.append("abacaxi")
print(frutas)
print(frutas[0])
print(frutas[1])
frutas.pop()
print(frutas)
```



9

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

c) VARIÁVEIS E TIPOS

- ✓ Dependendo do tipo de dado, diferentes operadores podem ser suportados.
- ✓ Há operadores aritméticos, relacionais, lógicos, etc.
- ✓ Cada operador é adequado a um tipo específico de dado e possui um resultado também que depende do tipo de dado utilizado na operação

10

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

c) VARIÁVEIS E TIPOS

Exemplo 5: Criar e executar o programa seguinte.

```
# Operadores  
  
numero_1 = 1  
numero_2 = 2  
texto_1 = "abc"  
texto_2 = "def"  
numero_resultado = numero_1 + numero_2  
texto_resultado = texto_1 + texto_2  
logico_resultado = (numero_1 > numero_2)  
print(numero_resultado)  
print(texto_resultado)  
print(logico_resultado)
```

11

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

d) ENTRADA DE DADOS

- ✓ Para aplicações simples em Python, existe uma função, denominada de `input()` que permite que o programa receba entradas digitadas pelo usuário.
- ✓ Essa função permite que seja informada uma mensagem que servirá como instrução ao usuário.

12

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

d) ENTRADA DE DADOS

Exemplo 6: Criar e executar o programa a seguir.

```
# Entrada de dados

nome = input("Digite seu nome: ")
idade = int(input("Digite sua idade: "))
dados = {"nome":nome, "idade":idade}
print(dados)
print("Nome: ", dados['nome'])
print("Idade: ", dados['idade'])
```

Ln: 10 Col: 0

13

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

e) ESTRUTURAS DE CONTROLE

- ✓ Assim com outras linguagens de programação, Python possui estruturas do tipo *if-else* e também uma variação do tipo *if-elif-else*, que permite seleções múltiplas.
- ✓ Blocos pertencentes aos *ifs*, *elifs* ou *elses* são delimitados por : e identificados pela **indentação (recuo)**.

14

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

e) ESTRUTURAS DE CONTROLE

Exemplo 7: Criar e executar o programa seguinte.

```
# Estrutura if-elif-else
x = 1
y = 3
if x < 1:
    print('x menor do que 1')
else:
    print('x maior ou igual a 1')

if y > 3:
    print('y é maior do que 3')
elif y==3:
    print('y é igual a 3')
else:
    print('y?? ')
```

15

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

e) ESTRUTURAS DE CONTROLE

Exercício: Considere o seguinte problema: Determinar se uma aeronave deve ou não entrar em manutenção. O programa deve ler os seguintes dados: Fabricante da aeronave (String), número de horas de voo desde a última manutenção. Os dados devem ser armazenados em um dicionário. Uma mensagem do tipo: “EFETUAR MANUTENÇÃO” + NOME DO FABRICANTE deve ser exibida nos seguintes casos:

1. Fabricante Boeing, horas de voo desde a última manutenção ≥ 250 horas.
2. Fabricante Bombardier, horas de voo desde a última manutenção: ≥ 180 horas
3. Demais casos: 150 horas.

16

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

f) ESTRUTURAS DE REPETIÇÃO

- ✓ Repetições em Python podem ser efetuadas utilizando-se as estruturas **while** e **for**
- ✓ Enquanto o bloco do *while* é executado até que sua condição se torne falsa, para o caso do *for*, tipicamente há uma inicialização, um incremento e uma checagem implícitos no comando.
- ✓ Em ambos os casos, a delimitação do bloco onde ocorre a iteração se dá por : e pela indentação.

17

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

f) ESTRUTURAS DE REPETIÇÃO

Exemplo 8: Criar e executar o programa a seguir.

```
# Estruturas de repetição
x = 0
while ( x < 10 ):
    print(x, end='; ')
    x = x+1
print()
for y in range(1,10,1):
    print(y)

frutas = ['abacate', 'abacaxi']
for fruta in frutas:
    print(fruta)
```

18

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES



f) ESTRUTURAS DE REPETIÇÃO

Exercício: Modifique o programa que é utilizado para determinar a manutenção de modo que:

1. Ignore o fato dos caracteres em caixa alta ou caixa baixa;
2. Seja executado indefinidamente até que o usuário digite “nenhum” para o fabricante da aeronave. Nesse caso, deve-se imprimir a mensagem: “saindo do programa”. Use dentro de um if, a instrução break para sair do loop.

19

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

g) FUNÇÕES

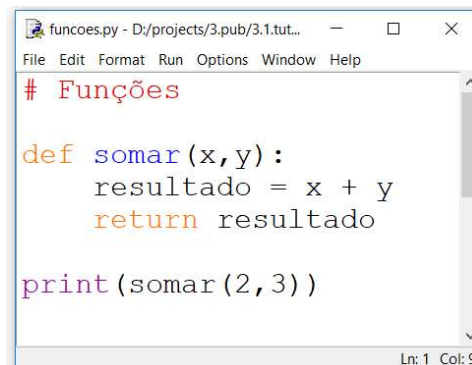
- ✓ Em Python funções são criadas através da utilização da palavra reservada **def**, seguida pelo nome da função, seus parâmetros entre parêntesis, seguido de :
- ✓ Uma função que retorna um valor faz isso através da utilização da palavra reservada **return**.

20

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

g) FUNÇÕES

Exemplo 9: Criar a seguinte função que retorna a soma de dois números.



```

# Funções

def somar(x, y):
    resultado = x + y
    return resultado

print(somar(2, 3))
  
```

21

4. DESENVOLVIMENTO DE PROGRAMAS SIMPLES

g) FUNÇÕES

Exercício: Crie uma função que receba como entrada duas resistências, r_1 e r_2 e que forneça o valor da resistência equivalente em série ($r_1 + r_2$) ou em paralelo $(r_1 \times r_2) / (r_1 + r_2)$. O valor equivalente deve depender de uma terceira entrada chamada montagem. “p” para paralelo e “s” para série.

A função deve ter a seguinte “assinatura”:
`calcula_resistencia(r1, r2, montagem)`

22

FIM DA PARTE 2!!