Desempenho de Negócios

Aula 02 – Introdução à linguagem Python

Renato Rodrigues Oliveira da Silva
renato.silva@impacta.edu.br

Sumário

- Histórico e Características
- Instalação
- Sintaxe
- Tipos de Dados e Operadores
- Controle de Fluxo
- Funções

Histórico

- Criada nos anos 1990 pelo holandês Guido van Rossum
- Nome inspirado no show de tv Monty Python
- Linguagem interpretada de propósito geral
  - Programa definido em um script, em oposição a um programa executável
- Linguagem de alto nível
  - Foco em facilidade de leitura e produtividade

O que dá pra fazer com Python?

- Ciência
  - Estatística, Manipulação de Dados
- Administração de sistemas
  - Unix, Linux
- Desenvolvimento web
  - Django
- Desenvolvimento de Interfaces Gráficas
  - PyQt, wxPython, Kivy

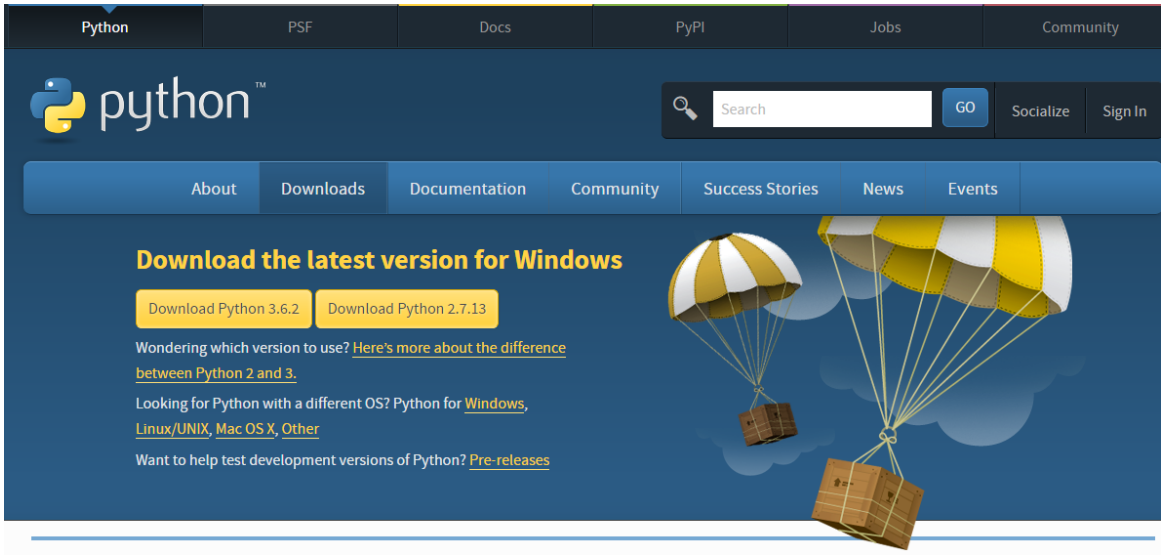
Por que usar Python?

- Linguagem dinâmica e orientada a objetos
- Vasta coleção de bibliotecas
- Gerenciamento automático de memória
- Portabilidade
  - Basta possuir um interpretador Python instalado no sistema
- Facilidade de uso e aprendizado

Quem usa Python?

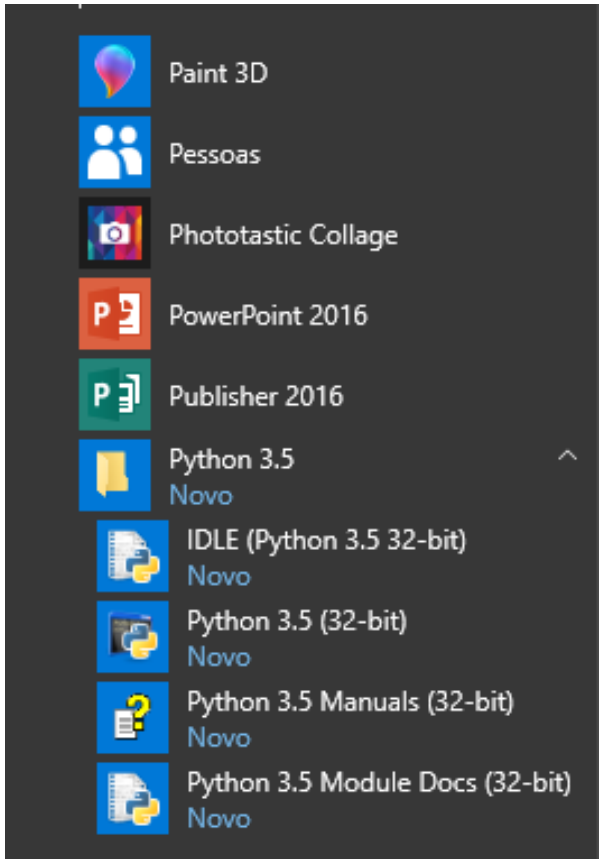
Instalação

- Pré-instado na maioria dos sistemas Unix
- Para o Windows, fazer o download no site **http://www.python.org/downloads**
  - Versão mais atual em Agosto de 2017: 3.4.7



Instalação

- Acompanha o ambiente de desenvolvimento IDLE



Instalação

- Também existem ambientes de desenvolvimento online
  - https://repl.it/languages/python3

Ambiente Online

- **repl.it**
  - https://repl.it/languages/python3

Ambiente Mobile

- QPython3

Sintaxe

- Código de Exemplo

```
print (“Hello World!”)
X = 10 – 3
Y = “Palavra”
if X == 7:
    print(“Concatenação” + y)
B = input (“Digite o seu nome”)
```

Sintaxe

- A quantidade de espaços (identação) é utilizada para definir blocos lógicos
  - Usadas em fluxo de controle, estruturas de repetição, etc.
  - A convenção é utilizar 4 caracteres de espaço
- A criação de uma variável acontece na primeira atribuição de um valor
  - Atribuição é **=**
  - Comparação é **==**

Sintaxe

- Comentários são definidos pelo caractere #
- Exemplo:

```
X = 10      #isso é um comentário
```

- Comentários são utilizados para tornar o código mais claro, e tornar a leitura mais fácil

Tipos de Dados

- Strings
  - X = “Isso é um String”
- Números Inteiros
  - Y = 10
  - Y = Int(“10”)
- Números de ponto flutuante
  - Z = 5.4
  - Z = Float(“5.4”)

Tipos de Dados

- Tipo nulo
  - Y = None
- Listas
  - X = [1,2,3,4, “Um”, “Dois”]
  - X.append(1)
  - Para acessar um elemento, usa-se os colchetes e indica a posição: X[1]
  - Para o tamanho da lista usa-se a função len: len(X)

Tipos de Dados

- Dicionários
  - Armazena valores indexados por chaves
  - Forma conveniente de armazenamento
  - X = {“Um” : 1, “Dois” : 2, “Tres” : 3}
- Booleanos
  - True
  - False

Operadores

- Aritméticos

```
a = 10
b = a+1     # 11
c = a-1     # 9
d = a*2     # 20
e = a/2     # 5
f = a%3     # 1
g = a**2    # 100
```

Operadores

- Manipulação de Strings

  Variavel = “Um” + “Dois” #UmDois

  A = 10

  B = 20

  C = “{}{}”.format(A,B) #1020

Operadores

- Lógicos

  A **and** B # e lógico

  A **or** B # ou lógico

  not A # negação

  (A and not (A and B)) # composição

Operadores

- Comparação Aritmética

  A > B

  A >= B

  A < B

  A <= B

  A == B

  A != B

Controle de Fluxo

```
A = 30
if A > 20:
    print (“Maior”)
else:
    print (“Menor”)
```

Saída: Maior

Controle de Fluxo

- Repetição **for**

```
for x in range(10):  # 0 a 9
    print (x)

marcas = [“Chevrolet”, “Ford”, “Volkswagen”]
for marca in marcas:
    print (marca)
```

Controle de Fluxo

- Enquanto **while**

```
x = 0
while x < 100:
    print (x)
    x = x + 1
```

Funções

- Facilitam a codificação de tarefas repetitivas
- Tornam o código menor e mais fácil de entender

```
def minha_função():
    """Documentação da função"""
    print("Hello World!")
```

Funções – Argumentos

- Posicional

```
def add(x, y):
    return x + y
```

- Palavra-chave

```
def alerta(mensagem="Alerta!"):
    print (mensagem)
```

- Posicional + palavra-chave

```
def alerta(texto, prefixo=""):
    print ("{}{}".format{prefixo, texto})
```

Importação de módulos

- Módulos permitem que o código seja isolado e re-utilizado
  - Conforme programas evoluem, as partes podem ser separadas em diferentes arquivos
  - As partes podem ser re-utilizadas por outros programas
- **import** nome_do_módulo
  - Permite utilizar as funções e variáveis do módulo
  - Uso: nome_do_módulo.função

Importação de módulos – Exemplos

- import datetime

```
datetime.date.today()   # data atual
```

Exercícios

1. Crie um programa para solicitar ao usuário digitar três números
   - Exiba o maior valor digitado
   - Exiba a média dos três valores

Exercícios

2. Crie um programa para pedir ao usuário entrar com o valor de largura e altura de um retângulo. Depois calcule e exiba da área na tela.
   - Área do Retângulo = Largura * Altura

Exercícios

3. Crie um programa para calcular o IMC (Índice de Massa Corpórea)
   - O usuário deve digitar o peso e a altura
   - O IMC é calculado pela expressão Peso / Altura^2
   - Informar a situação do usuário:
     - IMC < 18,5: Peso Baixo
     - 18,5 < IMC < 24,99: Normal
     - IMC > 25: Sobrepeso

Exercícios

4. Faça um Programa que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês.
   - Calcule o total do seu salário no referido mês, sabendo-se que são descontados:
     - 11% para o Imposto de Renda
     - 8% para o INSS
   - Exibir o valor do salário bruto, valor líquido e de cada desconto

Exercícios

5. Faça um programa que imprima na tela os números de 1 a 20, um abaixo do outro. Depois modifique o programa para que ele mostre os números um ao lado do outro.
   - Obs: O caractere de quebra de linha em Python é o “\n”.

Exercícios

6. Faça um programa que leia do usuário 5 números inteiros. Guarde-os em uma lista e mostre-os.

Exercícios

7. Faça um Programa que leia e armazene duas listas com 10 elementos cada. Gere uma terceira lista de 20 elementos, cujos valores deverão ser compostos pelos elementos intercalados das duas outras listas.