



# Introdução ao Python

Aprendendo a Programar com Python

Módulo 01

**Augusto Mathias Adams**

29/05/2023





O Nivelamento de ***Conceitos de Programação - Introdução ao Python*** tem como objetivo introduzir os participantes à linguagem de programação *Python*, fornecendo as habilidades básicas para escrever programas funcionais, compreender conceitos fundamentais e promover a resolução de problemas. Ao final do nivelamento, os alunos estarão preparados para aplicar seus conhecimentos em projetos pessoais ou profissionais.



# Sumário

## 1 Ementa

- ▶ Ementa
- ▶ Introdução ao Python
  - Ementa
  - Conceitos De Programação
  - Algoritmos
- ▶ Linguagem Python
  - O que é *Python*
  - Sintaxe
  - Estruturas de Controle
  - Estruturas de Dados
  - Funções em Python
  - Aplicativo em *Python*
- ▶ Exercícios



# Pré-requisitos

## 1 Ementa

Requisitos mínimos:

- Nenhum conhecimento prévio de programação é necessário.
- Um computador com **Python** instalado.

É desejável:

- *git* instalado (para baixar os exercicios)
- Conhecimentos mínimos de *Docker* (será ministrado ao longo do curso)



# Conteúdo do Nivelamento

## 1 Ementa

O conteúdo deste nivelamento será dividido em 5 partes:

- **Módulo 1:** introdução ao *Python* e princípios básicos de algoritmos.
- **Módulo 2:** introdução ao uso de pacotes e programação procedural com *Python* e introdução ao pacote *camera-discovery*.
- **Módulo 3:** introdução à programação orientada a objetos e conceitos de banco de dados.
- **Módulo 4:** introdução ao *Django - Framework* de desenvolvimento web usando python.
- **Módulo 5:** produção de um sistema de monitoramento de câmeras usando os pacotes *camera-discovery* e *Django*.



# Conteúdo do Nivelamento

## 1 Ementa

Ao final de cada módulo, será proposta uma lista de exercícios de fixação do conteúdo visto em sala de aula.



# Conteúdo do Nivelamento

## 1 Ementa

### Bibliografia:

- *Beginning Programming with Python for Dummies*
- *Algorithms for Dummies*
- *Django for Beginners*
- *Django Book*
- *Django ORM Cookbook*
- *The Definitive Guide to Django*
- *Apostila Módulo 01*



# Sumário

## 2 Introdução ao Python

### ► Ementa

### ► Introdução ao Python

Ementa

Conceitos De Programação

Algoritmos

### ► Linguagem Python

O que é *Python*

Sintaxe

Estruturas de Controle

Estruturas de Dados

Funções em Python

Aplicativo em *Python*

### ► Exercícios





# Objetivos do Módulo

## 2 Introdução ao Python

- Do que se trata **programar**.
- O que significa o termo **algoritmo**.
- Onde encontrar informações sobre como instalar o **Python**.
- **Sintaxe e Estrutura de Dados**.
- **Funções em Python**.
- Primeira Aplicação em **Python**.
- Comentários na sua primeira aplicação.



# Conceitos de Programação

## 2 Introdução ao Python

Programar é:

- Ato de escrever um conjunto de instruções ou algoritmos que um computador pode executar para realizar uma tarefa específica.
- Criação de código em uma linguagem de programação que segue uma sintaxe e estrutura definidas.
- Requer habilidades lógicas e analíticas para que as instruções sejam escritas de forma clara e precisa.



# Conceitos de Programação

## 2 Introdução ao Python

Programar não é:

- Falta de planejamento e análise antes de começar a escrever o código.
- Ausência de comentários explicativos.
- Uso excessivo de linhas de código redundantes ou desnecessária.
- Mistura de diferentes estilos e convenções de codificação.
- Dificuldade em manter e fazer modificações no código devido à falta de estrutura.



# Algoritmo

## 2 Introdução ao Python

### Definições:

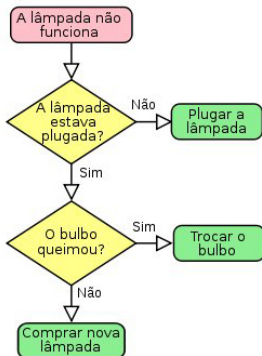
- **Algoritmo:** sequência de passos usada para resolver um problema, fornecendo uma solução específica.
- Para ser considerado algoritmo:
  - **Finito:** eventualmente resolve o problema.
  - **Bem definido:** série de passos precisa e etapas compreensíveis.
  - **Efetivo:** resolve todos os casos do problema para o qual foi definido.

**Lembre-se:** seguindo os passos da receita, conseguimos fazer o bolo!



# Algoritmos

## 2 Introdução ao Python



### Medicamento Anvisa®

#### Paracetamol

##### APRESENTAÇÕES

Comprimidos revestidos de  
- 500 mg em embalagem com 20 ou 200 comprimidos.  
- 750 mg em embalagens com 20 ou 200 comprimidos.

##### USO ORAL

USO ADULTO ACIMA DE 12 ANOS

##### COMPOSIÇÃO

Medicamento Anvisa® 500 mg:  
Cada comprimido revestido contém 500 mg de paracetamol.  
Excipientes: ácido estéarico, amido pré-gelatinizado, hipromelose, macrogol e povidona.

Medicamento Anvisa® 750 mg:  
Cada comprimido revestido contém 750 mg de paracetamol.  
Excipientes: ácido estéarico, amido pré-gelatinizado, hipromelose, macrogol e povidona.

##### 1. PARA QUÊ ESTE MEDICAMENTO É INDICADO?

Medicamento Anvisa® é indicado para o tratamento de febre e de dores leves a moderadas, de adultos, tais como: dores associadas a gripes e resfriados comuns, dor de cabeça, dor de dente, dor nas costas, dores associadas a artrites e cólicas menstruais.

##### 2. COMO ESTE MEDICAMENTO FUNCIONA?

Medicamento Anvisa® reduz a febre atuando no centro regulador da temperatura no Sistema Nervoso Central (SNC) e diminui a sensibilidade para a dor. Seu efeito tem início 15 a 30 minutos após a administração oral e permanece por um período de 4 a 6 horas.



# Algoritmos

## 2 Introdução ao Python

Algoritmos comuns (implementados na maioria das linguagens):

- Algoritmos de Busca
- Algoritmos de Classificação e Ordenação
- Algoritmos de Transformação de Dados
- Algoritmos de Agendamento
- Algoritmos de Criptografia
- Geração de números pseudo-aleatórios



# Sumário

## 3 Linguagem Python

- ▶ Ementa
- ▶ Introdução ao Python
  - Ementa
  - Conceitos De Programação
  - Algoritmos
- ▶ Linguagem Python
  - O que é *Python*
  - Sintaxe
  - Estruturas de Controle
  - Estruturas de Dados
  - Funções em Python
  - Aplicativo em *Python*
- ▶ Exercícios



# O que é Python

## 3 Linguagem Python

Do que se trata?

- **Definição:** Linguagem de programação simples e versátil
- **Características:**
  - Multiplataforma
  - Multiparadigma
  - Linguagem Interpretada





# O que é *Python*

## 3 Linguagem Python

Não posso gerar executável a partir do ***Python***? Pode.....

- Py2Exe
- PyInstaller



# Instalando o *Python*

3 Linguagem Python

Guia passo a passo para a instalação do *Python*

- *Apostila de Instalação*



# Estrutura e Sintaxe

## 3 Linguagem Python

A sintaxe do *Python* é a forma como o código em *Python* é escrito e estruturado.

- O código escrito em *Python* é estruturado em blocos;
- indentação faz parte da estrutura;
- sem necessidade de caracteres de abertura e fechamento de Blocos;
- sem necessidade de terminador de linha;
- comentários simples e multilinha;
- Variáveis são utilizadas para armazenar valores e sua atribuição é feita através de um sinal de igualdade.



# Programa Simples Escrito em Python

## 3 Linguagem Python

Exemplo:

---

```
# comentário de uma linha
# atribuindo o valor 5 a uma variável x
x = 5
if x > 5:
    # inicio de um bloco
    print("x é maior que 5")
    # fim de um bloco
else:
    # inicio de outro bloco
    print("x é menor ou igual a 5")
    # fim de outro bloco
```

---



# Comentários Multilinha

## 3 Linguagem Python

Exemplo de comentário multilinha:

---

```
"""  
Introdução ao Python - CICC  
@author Augusto Mathias Adams <augusto.mathias@sesp.pr.gov.br>  
MIT License  
.....  
"""
```

---



# Estruturas de Controle

## 3 Linguagem Python

As estruturas de controle em **Python** são usadas para controlar o fluxo de execução de um programa. Elas permitem que você tome decisões com base em condições e repita a execução de um bloco de código várias vezes.



# Estruturas Condicionais

## 3 Linguagem Python

- **if:** Executa um bloco de código se uma condição for verdadeira.
- **elif:** Permite testar condições adicionais se as condições anteriores forem falsas.
- **else:** Executa um bloco de código se todas as condições anteriores forem falsas.



# Estruturas Condicionais

## 3 Linguagem Python

Exemplo:

---

```
x = 5
if x > 0:
    print("x é positivo")
elif x == 0:
    print("x é igual a zero")
else:
    print("x é negativo")
```

---





# Estruturas de Repetição

## 3 Linguagem Python

- **for:** Itera sobre uma sequência (como uma lista, tupla ou string) ou um objeto iterável por um número específico de vezes.
- **while:** Executa um bloco de código repetidamente enquanto uma condição for verdadeira.



# Estruturas de Repetição

## 3 Linguagem Python

Exemplo:

---

```
# loop for  
for i in range(5):  
    print(i)
```

```
x = 10  
# loop while  
while x > 0:  
    print(x)  
    x -= 1
```

---



# Controle de Loop

## 3 Linguagem Python

- **break:** Encerra imediatamente o loop em que está sendo executado.
- **continue:** Pula para a próxima iteração do loop, ignorando o restante do código dentro do bloco do loop.



# Controle De Loop

## 3 Linguagem Python

Exemplo:

---

```
# exemplo de break
for i in range(10):
    if i == 5:
        break
    print(i)

# exemplo de continue
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)
```

---



# Tratamento de Exceções

## 3 Linguagem Python

- **estrutura try .... except:** utilizada para lidar com exceções e tratar erros de forma controlada. Ela permite que você execute um bloco de código suscetível a erros e defina como lidar com esses erros, evitando que o programa seja interrompido abruptamente.



# Tratamento de Exceções

## 3 Linguagem Python

Exemplo:

---

```
try:
    numero = int(input("Digite um número: "))
    resultado = 10 / numero
    print("O resultado é:", resultado)
except ValueError:
    print("O valor digitado não é um número válido.")
except ZeroDivisionError:
    print("Não é possível dividir por zero.")
else:
    print("Nenhum erro ocorreu.")
finally:
    print("Fim da execução.")
```

---



# Tratamento de Exceções

## 3 Linguagem Python

- O código dentro do bloco `try` é executado normalmente.
- Se ocorrer uma exceção no bloco `try`, o fluxo de execução é interrompido e o **Python** procura pelo bloco `except` correspondente à exceção lançada.
- Se o tipo de exceção lançada for correspondente a algum bloco `except`, o código dentro desse bloco é executado.
- Se a exceção lançada não corresponder a nenhum bloco `except`, ela será propagada para níveis superiores na pilha de chamadas ou tratada por um bloco `except` mais geral.
- O bloco `else` é opcional e é executado se nenhum erro ocorrer no bloco `try`.
- O bloco `finally` é opcional e sempre é executado, independentemente de ocorrer uma exceção ou não. É usado para executar código de limpeza ou finalização.



# Estruturas de Dados

## 3 Linguagem Python

- **Listas(list):** coleção ordenada e mutável de elementos, que podem ser de diferentes tipos. Os elementos de uma lista são acessados através de índices, onde o primeiro elemento tem índice 0.
- **Dicionários(dict):** coleção de pares chave-valor, onde cada chave é única e mapeada a um valor correspondente. Os dicionários são úteis para armazenar informações relacionadas e acessá-las de forma eficiente através das chaves.





# Estruturas de Dados

## 3 Linguagem Python

Exemplo:

---

```
# exemplo de lista
frutas = ['maçã', 'banana', 'laranja']
print(frutas[0])  # saída: 'maçã'

# exemplo de dicionário
contatos = {'João': '123456789', 'Maria': '987654321'}
print(contatos['João'])  # saída: '123456789'
```

---



# Estruturas de Dados

## 3 Linguagem Python

- **Tuplas(tuple):** Semelhante a uma lista, uma tupla é uma coleção ordenada de elementos. A diferença é que as tuplas são imutáveis, ou seja, seus elementos não podem ser alterados após a criação.
- **Conjuntos(sets):** coleção não ordenada de elementos únicos. Os conjuntos não permitem elementos duplicados e suportam operações como união, interseção e diferença.



# Estruturas de Dados

## 3 Linguagem Python

Exemplo:

---

```
# exemplo de tupla  
ponto = (3, 4)  
print(ponto[0]) # saída: 3
```

```
# exemplo de conjunto  
numeros = {1, 2, 3, 4, 5}  
numeros.add(6)  
print(numeros) # saída: {1, 2, 3, 4, 5, 6}
```

---



# Estruturas de Dados

## 3 Linguagem Python

- **Strings:** sequência imutável de caracteres. Elas são usadas para armazenar e manipular texto em **Python**. As strings podem ser acessadas por índices e suportam várias operações, como concatenação e fatiamento (*slicing*).



# Estruturas de Dados

## 3 Linguagem Python

Exemplo:

---

```
# exemplo de string  
mensagem = "Olá, mundo!"  
print(mensagem[4]) # saída: ' '
```

---



# Funções em Python

## 3 Linguagem Python

- **Funções:** blocos de código reutilizáveis que executam uma tarefa específica. Elas são usadas para agrupar um conjunto de instruções relacionadas e podem receber argumentos (valores de entrada) e retornar resultados (valores de saída).
- **Vantagens:** Modularidade, Reutilização de Código, Legibilidade, Manutenção, Testabilidade, Organização.



# Funções em Python

## 3 Linguagem Python

Estrutura de uma função:

- **Definição:** Uma função é definida usando a palavra-chave `def`, seguida pelo nome da função e parênteses contendo os argumentos, se houver.
- **Parâmetros:** Os parâmetros são variáveis que recebem os valores passados para a função quando ela é chamada.
- **Corpo da função:** bloco de código indentado que contém as instruções a serem executadas quando a função é chamada.
- **Retorno de valores:** Uma função pode retornar um valor usando a palavra-chave `return`. Isso permite que a função forneça um resultado para o código que a chamou.



# Funções em Python

## 3 Linguagem Python

Exemplo:

---

```
# Definindo uma função que implementa a operação de soma
def soma(a, b):
    resultado = a + b
    return resultado

# Chamando a função e imprimindo o resultado
resultado_soma = soma(5, 3)
print(resultado_soma)
```

---





# Funções em Python

## 3 Linguagem Python

Exemplo com tipo de retorno definido:

---

```
# Definindo uma função que implementa a operação de soma
def soma(a, b) -> float:
    resultado = a + b
    return resultado

# Chamando a função e imprimindo o resultado
resultado_soma = soma(5, 3)
print(resultado_soma)
```

---



# Funções em Python

## 3 Linguagem Python

- **Funções Anônimas:** são funções em linha de código que não possuem um nome definido. Elas são úteis quando você precisa de uma função simples e de curta duração.
  - São frequentemente usadas em combinação com outras funções, como *map()*, *filter()*, e *reduce()*, para realizar operações em coleções de dados de forma concisa.



# Funções em Python

## 3 Linguagem Python

Exemplo:

---

```
# definindo uma função lambda
```

```
quadrado = lambda x: x**2
```

```
# Chamando a função lambda
```

```
resultado = quadrado(5)
```

```
print(resultado)
```

---



# Aplicativo em Python

## 3 Linguagem Python

Tendo como base o conhecimento adquirido até agora, vamos implementar uma calculadora com as 4 operações básicas:

- Implemente uma função para cada uma das 4 operações da calculadora (soma, subtração, multiplicação e divisão)
- implemente o *loop* principal utilizando a função `input` nativa do **Python** e um *loop* *while* com a condição sempre verdadeira. É um *loop* infinito, porém, não se preocupe com isto agora.
- Não tem idéia de como fazer? Temos uma colinha especial em: **Calculadora de Padeiro**



# Sumário

4 Exercícios

- ▶ Ementa
- ▶ Introdução ao Python
  - Ementa
  - Conceitos De Programação
  - Algoritmos
- ▶ Linguagem Python
  - O que é *Python*
  - Sintaxe
  - Estruturas de Controle
  - Estruturas de Dados
  - Funções em Python
  - Aplicativo em *Python*
- ▶ Exercícios



# Exercício 1 - Algoritmos

4 Exercícios

Receita de Bolo da Vovó:

- Pegue uma receita de bolo, ou de qualquer prato que goste (roubar uma receita da esposa serve)
- leia atentamente a receita.
- descreva utilizando um fluxograma passo a passo a confecção da receita
- Dica:
  - A receita geralmente é dividida em duas partes: ingredientes e modo de fazer. Inclua os ingredientes como variáveis de entrada. O modo de fazer é, essencialmente, o algoritmo. Divida em quantas partes achar necessário.



## Exercício 2 - Algoritmos

4 Exercícios

Receita de Bolo da Vovó em *Python*:

- Crie uma função para cada item do algoritmo definido no exercício anterior.
- Crie uma função que gerenciará os passos de execução do algoritmo.
- Crie uma chamada de função ao gerenciador que criou e exiba a saída do programa.
- **Dica:** copie a estrutura da minha receita contida em [Receita de Bolo da Vovó](#)



## Exercício 3 - Algoritmos

4 Exercícios

Implemente um sistema de recomendação para solução de problemas, estilo do algoritmo da lâmpada.

- Não vale utilizar o mesmo exemplo.
- Encontre um problema que se resolva em 3 passos na sua casa.
- Crie um fluxograma de solução do problema.
- Implemente utilizando funções em **Python**, ao estilo do exercício 1.





## Exercício 4 - Algoritmos

4 Exercícios

O que falta na bula de remédios para se tornar um algoritmo? Comente pelo menos 2 casos aplicáveis.



# Introdução ao Python *Agradeço a atenção!*

*Questionamentos?*