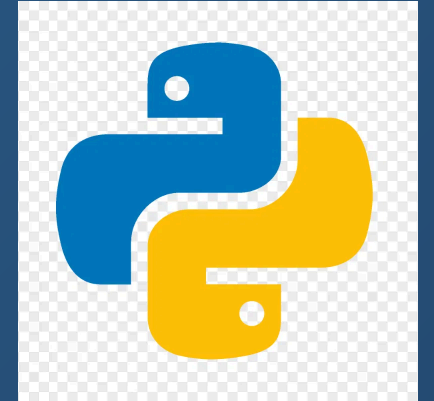






# Introdução ao Python



## O que é Python?

Python é uma linguagem de programação de **alto nível**, **interpretada** e **orientada a objetos**. Foi criada por **Guido van Rossum** em 1991.

## Aplicações

-  Desenvolvimento Web
-  Desenvolvimento de Software
-  Ciência de Dados
-  Automação e Scripting

## Popularidade

Python é consistentemente classificada entre as **linguagens de programação mais populares** do mundo, segundo o índice TIOBE e Stack Overflow.

# Por que usar Python?

## Sintaxe Simples e Legível

```
# Python
for i in range(5):
    print("Número:", i)
```

## Multiplataforma

Funciona em **Windows**, **macOS**, **Linux** e outros sistemas operacionais.

## Produtividade

- ✓ Menos linhas de código
- ✓ Desenvolvimento rápido
- ✓ Grande biblioteca padrão

## Versatilidade

Suporta múltiplos paradigmas:

- ✓ **Procedural**
- ✓ **Orientado a objetos**
- ✓ **Funcional**

## Comunidade Ativa

Ampla comunidade de desenvolvedores, extensa documentação e milhares de bibliotecas disponíveis através do **PyPI** (Python Package Index).

# Primeiros Passos com Python

## 📁 Instalação

- ✓ Baixe em [python.org](https://python.org)
- ✓ Disponível para Windows, macOS e Linux
- ✓ Verifique a instalação com `python --version`

## ➤ Executando Python

Modo interativo (REPL):

```
# No terminal/cmd
$ python
>>> print("Olá, Mundo!")
Olá, Mundo!
>>> exit()
```

```
activate.csh
activate.fish
easy_install
easy_install-3.4
pip
pip3
pip3.4
python -> python3
python3 -> /usr/bin/python3
include
lib
├── python3.4
├── python-wheels
lib64 -> lib
pyenv.cfg
test.py
6 directories, 12 files
```

## 📄 Primeiro Programa

Arquivo `hello.py`:

```
# Meu primeiro programa
print("Olá, Mundo!")
```

Execute com: `python hello.py`

## 🔧 Ferramentas de Desenvolvimento

📄 **Editores:** VS Code, PyCharm

🧪 **Notebooks:** Jupyter, Google Colab

✍️ **Online:** W3Schools, Replit

# Sintaxe Básica e Variáveis

## Indentação

```
# Indentação é obrigatória
if 5 > 2:
    print("Cinco é maior que dois!")

# Erro de sintaxe:
if 5 > 2:
print("Isso causará um erro")
```

## Variáveis

```
# Criação de variáveis
x = 5
y = "Olá"
print(x, y)

# Tipagem dinâmica
x = "Agora sou uma string"
print(x)
```

## Comentários

```
# Isto é um comentário de linha única
print("Hello, World!") # Comentário após código

"""
Este é um comentário
de múltiplas linhas
"""
```

## Verificando Tipos

```
# Função type()
x = 5
y = "Olá"

print(type(x)) # <class 'int'>
print(type(y)) # <class 'str'>
```

# Tipos de Dados e Operadores

## Tipos de Dados Básicos

```
# Tipos de dados em Python
texto = "Olá, Python!" # str
inteiro = 42 # int
decimal = 3.14159 # float
complexo = 1+2j # complex
booleano = True # bool

# Verificando tipos
print(type(texto)) # <class 'str'>
print(type(inteiro)) # <class 'int'>
```

## Tipos de Coleções

```
# Coleções em Python
lista = ["maçã", "banana", 42] # list
tupla = ("maçã", "banana", 42) # tuple
conjunto = {"maçã", "banana", "laranja"} # set
dicionario = {
    "nome": "Python",
    "ano": 1991
} # dict
```

## Operadores Aritméticos

```
# Operadores aritméticos
a = 10
b = 3
print(a + b) # Adição: 13
print(a - b) # Subtração: 7
print(a * b) # Multiplicação: 30
print(a / b) # Divisão: 3.3333...
print(a % b) # Módulo: 1
print(a ** b) # Exponenciação: 1000
print(a // b) # Divisão inteira: 3
```

## Operadores de Comparação

```
# Operadores de comparação
x = 5
y = 10
print(x == y) # Igual a: False
print(x != y) # Diferente de: True
print(x > y) # Maior que: False
print(x < y) # Menor que: True
print(x >= y) # Maior ou igual: False
print(x <= y) # Menor ou igual: True
```

# Estruturas de Dados

## ☰ Listas

```
# Criando e manipulando listas
frutas = ["maçã", "banana", "laranja"]
frutas.append("uva") # Adiciona item
frutas[1] = "morango" # Altera item
```

## 🌀 Sets

```
# Criando e manipulando sets
numeros = {1, 2, 3, 4, 5, 5} # Duplicata removida
numeros.add(6) # Adiciona item
numeros.remove(2) # Remove item
```

## 🔒 Tuplas

```
# Criando e acessando tuplas
cores = ("vermelho", "verde", "azul")
# cores[0] = "amarelo" # Erro!
```

## 🔑 Dicionários

```
# Criando e manipulando dicionários
pessoa = {
    "nome": "Ana",
    "idade": 25,
    "cidade": "São Paulo"
}
```

## ↔ Comparação de Estruturas

### Lista [ ]

- ✓ Ordenada
- ✓ Mutável

### Tupla ( )

- ✓ Ordenada
- ✗ Imutável

### Set { }

- ✗ Não ordenado
- ✗ Sem duplicatas

### Dict {k:v}

- ✓ Ordenado\*
- ✓ Chaves únicas

\* Ordenado a partir do Python 3.7

# Controle de Fluxo

## Estruturas Condicionais

```
# Estrutura if-elif-else
idade = 18
if idade < 18:
    print("Menor de idade")
elif idade == 18:
    print("Exatamente 18 anos")
else:
    print("Maior de idade")
```

## Loop While

```
# Loop while com break e continue
contador = 0
while contador < 5:
    contador += 1
    if contador == 3:
        continue # Pula o 3
    print(contador)
    if contador == 4:
        break # Para no 4
```

## Loop For

```
# Loop for com range()
for i in range(1, 5):
    print(i) # Imprime 1, 2, 3, 4
```

## Iterando Coleções

```
# Iterando sobre uma lista
frutas = ["maçã", "banana", "laranja"]
for fruta in frutas:
    print(fruta)
```

# Funções e Lambda

## Definindo Funções

```
# Definição de função
def saudacao(nome):
    return f"Olá, {nome}!"

# Chamando a função
mensagem = saudacao("Maria")
print(mensagem) # Olá, Maria!
```

## Argumentos

- ✓ **Posicionais:** ordem importa
- ✓ **Nomeados:** nome=valor
- ✓ **Padrão:** valor predefinido
- ✓ **Arbitrários:** \*args, \*\*kwargs

## Exemplo de função Python

## Funções Lambda

```
# Função lambda (anônima)
dobrar = lambda x: x * 2

print(dobrar(5)) # 10

# Lambda com múltiplos argumentos
somar = lambda a, b: a + b
print(somar(3, 4)) # 7
```

## Boas Práticas

- ✓ Nomes descritivos
- ✓ Uma função = uma tarefa
- ✓ Docstrings para documentação



# Programação Orientada a Objetos

## Classes e Objetos

```
# Definição de uma classe
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def apresentar(self):
        return f"Olá, sou {self.nome} e tenho {self.idade} anos."

# Criando um objeto
p1 = Pessoa("Ana", 25)
print(p1.apresentar())
```

## Herança

```
# Classe pai
class Animal:
    def __init__(self, nome):
        self.nome = nome

    def fazer_som(self):
        print("Som genérico")

# Classe filha
class Cachorro(Animal):
    def fazer_som(self):
        print("Au au!")
```

## Polimorfismo

```
# Usando polimorfismo
animais = [
    Cachorro("Rex"),
    Animal("Criatura")
]

for animal in animais:
    animal.fazer_som() # Método diferente para cada classe
```

## Encapsulamento

- ✓ **Público:** atributos e métodos normais
- ✓ **Protegido:** prefixo com um underscore (`_atributo`)
- ✓ **Privado:** prefixo com dois underscores (`__atributo`)

Python usa convenções de nomenclatura em vez de modificadores de acesso rígidos.

# Tópicos Avançados

## Módulos e Pacotes

```
# Importando módulos
import math
from datetime import datetime

# Usando módulos
raiz = math.sqrt(16) # 4.0
agora = datetime.now() # Data atual

# Criando seu próprio módulo
# Em arquivo utils.py:
# def saudacao(nome):
# return f"Olá, {nome}!"
```

## Tratamento de Exceções

```
try:
    numero = int(input("Digite um número: "))
    resultado = 10 / numero
    print(resultado)
except ValueError:
    print("Entrada inválida!")
except ZeroDivisionError:
    print("Não é possível dividir por zero!")
except Exception as e:
    print(f"Erro: {e}")
else:
    print("Operação bem-sucedida!")
finally:
    print("Processo finalizado.")
```

## Manipulação de Datas

```
from datetime import datetime, timedelta

# Data atual
agora = datetime.now()

# Adicionando dias
amanha = agora + timedelta(days=1)

# Formatando data
formatado = agora.strftime("%d/%m/%Y %H:%M")
```

## Manipulação de Arquivos

```
# Escrevendo em um arquivo
with open('arquivo.txt', 'w') as f:
    f.write('Olá, mundo!')

# Lendo um arquivo
with open('arquivo.txt', 'r') as f:
    conteudo = f.read()
    print(conteudo)
```

## JSON

```
import json

# Dict para JSON
dados = {'nome': 'João', 'idade': 30}
json_str = json.dumps(dados)
```

## RegEx

```
import re

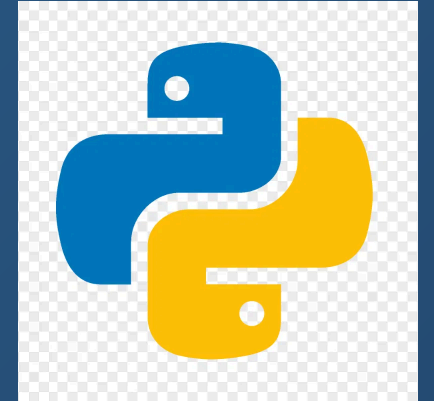
texto = "Email: contato@exemplo.com"
padrao = r'\S+@\S+\.\S+'
email = re.search(padrao, texto)
```

## PIP

```
# Instalar pacote
$ pip install numpy

# Listar pacotes
$ pip list
```

# Conclusão e Próximos Passos



## ✓ O que Aprendemos

- Fundamentos da linguagem
- Estruturas de dados
- Controle de fluxo
- Funções e Lambda
- Programação Orientada a Objetos
- Tópicos avançados

## 🏠 Próximos Passos

- ➔ Frameworks web (Django, Flask)
- ➔ Ciência de dados (Pandas, NumPy)
- ➔ Machine Learning (Scikit-learn, TensorFlow)
- ➔ Automação e DevOps

## 📖 Recursos para Continuar

- 🔗 Documentação oficial: [python.org](https://python.org)
- 🔗 Tutoriais: [W3Schools](#), [Real Python](#)
- 🔗 Cursos online: [Coursera](#), [Udemy](#)
- 🔗 Comunidade: [Stack Overflow](#), [GitHub](#)

## 💬 Lembre-se

"Python é uma linguagem poderosa e versátil que continuará a evoluir. A melhor maneira de aprender é **praticando** e **construindo projetos reais**."