Minicurso: Python básico para programadores

PRIMEIROS PASSOS

Acompanhe os Slides por aqui

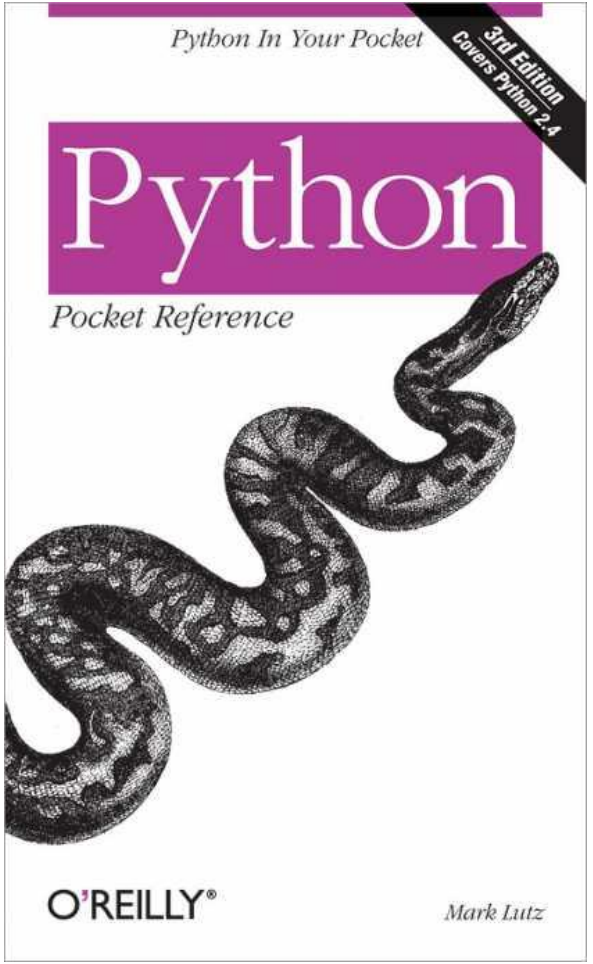
ou por aqui: https://dontpad.com/saet2025python

Criação do Python

- Criado por Guido van Rossum em 1982;
- Nessa época Guido trabalhava na Holanda no projeto da linguagem ABC;
- Necessidade de uma linguagem de fácil entendimento surgiu para os projetos;
- na época, Guido achava os programas escritos em C grandes e de difícil entendimento;

Criação do Python

- Nome em homenagem a um programa de TV(Monty Python’s Flying Circus);
- Depois de lançar o primeiro livro, a editora O’reilly estampou a serpente Python na capa;
- A ‘Python’ então acabou virando um dos símbolos da linguagem;



Guido van Rossum

- 2005 a 2012 trabalhou na Google;
- Em janeiro de 2013 começou a trabalhar com armazenamento em nuvem na dropbox;
- em 2019 anunciou sua aposentadoria;
- em 2020 anunciou o fim da aposentadoria e se juntou ao time de desenvolvedores da microsoft, onde trabalha até hoje;

Python

- Linguagem de propósito geral;
- fácil e intuitivo;
- multiplataforma;
- batteries included;
- código aberto;
- orientada a objeto;
- muitas bibliotecas;

Quem usa python?

Python é Lento?

Numa resposta curta...

Python é Lento?

Numa resposta curta...

NÃO

Hello World em outras linguagens

C	JAVA

`#include <stdio.h>`
 	
`int main(){`
`  printf("Hello World!\n");`
`  return 0;`
`}`
	`public class main{`
	`  public static void main(String[] args){`
	`  System.out.println("Hello World!");`
	`  }`
	`}`

Hello World em Python

Python

```
print("Hello World!\n")
```

Variáveis e tipos de dados

- Variáveis em python não são tipadas;
- Não é necessário declarar o tipo;
- Não é necessário declarar que é uma variável;

javascript	C	Python
`let a = 10;`	`int a = 10;`	`a = 10`

Variáveis e tipos de dados

```
media = 0
n1 = n2 = n3 = n4 = 0.0
nome, idade = "Carlos", 47
complexo = 2 + 3j
print("Hello World!\n")
print(type(media))
print(type(nome))
print(type(idade))
print(type(complexo))
```

Operadores Aritméticos

Operador	Operação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
//	Divisão inteira
%	Resto da divisão (Módulo)
**	Potenciação

Operadores de Comparação

Operador	Operação
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Operadores Lógicos (NOT)

A	not A
0	1
1	0

Operadores Lógicos (AND)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Operadores Lógicos (OR)

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Desvios Condicionais

- Delimitação de bloco feita por “:”
- É necessário indentação para o interpretador identificar o código que está dentro e fora do bloco;
- Não é necessário o uso de “()” para condições mais simples;

Desvios Condicionais

```
# Calcular média, se for maior ou igual a 7.0, aluno está aprovado, maior  igual a 5.0 em recuperação e caso seja menor que 5.0, reprovado.

n1 = float(input("Digite a nota1: "))
n2 = float(input("Digite a nota2: "))

media = (n1 + n2) / 2

if media >= 7.0:
    print("Aprovado!\n")
elif media >= 5.0:
    print("Recuperação!\n")
else:
    print("Reprovado!\n")

print("Sua média foi:", media, "\n")
```

Saída de dados formatadas (sem quebra de linha)

```
# desabilitando quebra de linha do print()
print("Imprime mensagem")
print("Imprime mensagem ...", end="")
print("Continua na mesma linha")
```

Saída de dados formatadas (função format)

```
nome = "Maria"
idade = 30

msg_formata = "O nome dela é {0} e ela tem {1}
anos.".format(nome, idade)

print(msg_formata)
```

Saída de dados formatadas (fstring)

```
# fstring para formatar também


nome = "Fernando"
peso = 80.5


print(f"Olá, meu nome é {nome} e tenho {peso}
kg.")
```

Saída de dados formatadas (fstring)

```
# É possível colocar expressões dentro de "{}""
também
a = 5
b = 10

print(f"A soma de {a} com {b} é: {a+b}")
```

Saída de dados formatadas (fstring)

```
# deixa o valor formatado em duas casas
decimais, arredondando pra cima, não trunca

valor = 258.1184844
print(f'O valor é: "{valor:.2f}"')
```

Laços de repetição (While)

```
# imprime os números de 1 a 10, num começa em 1, enquanto não for <= a 10, o laço continua repetindo
num = 1

while num <= 10:
    print(num)
    # num = num + 1
    num += 1
```

Laços de repetição (While)

```
# loop infinito, parada realizada usando a instrução
'break'
while True:
    nome = input('Digite um nome, caso deseje parar
digite "x": ')
    if nome == "x" or nome == "X":
        break
    print(f"Bem-vindo {nome}")
```

Laços de repetição (For)

```
lista = [1,2,3,4,5,6,7,8,9,10]


# para cada item na lista, faça:
for item in lista:
    print(item)
```

Laços de repetição (For)

```
palavra = "Boston"


# para cada letra na palavra, faça:
for letra in palavra:
    print(letra)
```

Laços de repetição (For)

```
# utiliza range() para rodar num número
predeterminado de vezes
# segundo argumento do range(m, n-1) vai até
n-1
for numero in range(1, 11):
    print(numero)
```

Laços de repetição (For)

```
# pode ser usado com um único argumento também,
nesse caso
# a contagem começa em 0 e vai até n-1 com
range(n)
for numero in range(1):
    print(numero)
```

Laços de repetição (For)

```
# ainda, a função range() pode ter um argumento
a mais, que decide
# o incremento do valor inicial ao valor final
# range(valor_inicial, valor_final, incremento)

for numero in range(1,10,2):
    print(numero)
```

Geração de números Aleatórios (Random)

```
import random


# gera um número inteiro aleatório entre 1 e 20
valor = random.randint(1, 20)
print(valor)
```

Geração de números Aleatórios (Random)

```
import random


# gera um valor aleatório entre 0 e 1
valor = random.random()
print(valor)
```

Geração de números Aleatórios (Random)

```
import random


# gera um valor fracionário aleatório entre
dois números
valor = random.uniform(1, 100)
print(valor)
```

Listas

```
# listas representam uma sequência de valores
notas1 = [4, 1, 7, 9, 10, 7, 3]
notas2 = [1, 10, 8, 5, 5, 3]
# concatena as listas
todas_notas = notas1 + notas2
# acesso a primeira posição da lista
print(todas_notas[0])
# acesso a última posição da lista
print(todas_notas[-1])
# acesso a penúltima posição da lista
print(todas_notas[-2])
```

Listas(Slicing)

```
# Slicing
# notação em python que indica uma faixa de valores a serem retornados
lista = [1,2,3,4,5,6,7,8,9,10]
print(lista[:]) # todos os valores da lista
print(lista[0:3]) # do primeiro elemento ao terceiro elemento
#o último elemento não é incluso
'lista[valor_inicial:valor_final-1]'
```

Mais funções de listas

```
lista = [1,2,3,4,5,6,7,8,9,10]
print(len(lista)) # imprime o tamanho da lista
print(sorted(lista)) # imprime a lista ordenada
print(sorted(lista, reverse=True)) # imprime a lista
ordenada ao contrário
print(sum(lista)) # imprime a soma dos valores da lista
print(min(lista)) # imprime o menor valor da lista
print(max(lista)) # imprime o maior valor da lista
```

Mais funções de listas

```
lista.append(13)  # adiciona o valor 13 ao final da
lista
lista.pop()  # remove o último valor da lista
lista.pop(3) # remove o elemento com índice 3
lista.insert(2,31) # insere no índice 2 o valor 31
print( 12 in lista) # retorna 'True' se 12 estiver
na lista
```

Tuplas

```
# tuplas são sequências de dados imutáveis
# são representadas por ()

tupla = (1, 234, 45, 123, 756, 146, 166, 0)
tupla[0] = 5 # Não é possível fazer atribuições
```

```
    tupla[0] = 5
TypeError: 'tuple' object does not support item assignment
```

Tuplas

```
# Todos os acessos que não modificam a tupla são
possíveis
tupla = (1, 234, 45, 123, 756, 146, 166, 0)
print(tupla[-1])# imprime última posição da tupla
print(tupla[0:3]) # imprime do indíce 0 ao indíce 2
print(tupla.count(1)) # conta a ocorrência do número 1 na
tupla
# ...
```

Tuplas

```
# É possível transformar uma tupla em uma lista
tupla = (1, 234, 45, 123, 756, 146, 166, 0)
lista = list(tupla)
print(lista)
```

Tuplas

```
# Também é possível transformar uma lista em
uma tupla
lista = [1231,15,15,6,4,2,7,11,45,90,1]
tupla = tuple(lista)
print(tupla)
```

Funções matemáticas internas

```
lista = [-1, -9, 10, 15, 4, 1, -19]
print(max(lista)) # maior valor da lista
print(min(lista)) # menor valor da lista
a = -5
b = 4
print(abs(a))  # módulo de a
print(pow(a, b))  # 625 pois faz (-5)^2 = 625
c = 2.7893452
print(round(c, 2)) # arredonda com duas casas
```

Módulo Math

```
import math


print(math.pi)  # constante pi
print(math.e)  # constante e
print(math.sqrt(80))  # imprime a raiz com várias casas
print(math.ceil(math.sqrt(80)))  # arredonda para o valor mais
alto
print(math.floor(math.sqrt(80)))  # arredonda para o valor mais
baixo
print(100 / math.inf)  # math.inf possui um valor muito grande,
logo o resultado é zero
```

Manipulação de Strings

```
frase = "O rato roeu a roupa do rei de Roma"
palavras = frase.split()  # split() separa a
frase em palavras e retorna uma lista
print(palavras)
```

Manipulação de Strings

```
email = input("Digite seu e-mail: ")
arroba = email.find("@") # retorna o índice do
@
usuario = email[0:arroba]
dominio = email[arroba + 1 :]
print(usuario)
print(dominio)
```

Manipulação de Strings

```
# pode achar qualquer substring dentro da
string maior
nome_produto = "Toalha de rosto algodão
banheiro"
print("Toalha"  in nome_produto)
```

Manipulação de Strings

```
texto = "tRÊs prAtos de TRIgo para 3 tiGres Tristes"
print(texto.capitalize())  # primeira letra do texto
em maíusculo
print(texto.title())  # toda primeira letra da palavra
em maíusculo
print(texto.lower())  # texto inteiro em minúsculo
print(texto.upper())  # texto inteiro em maíusculo
```

Manipulação de Strings

```
# É possível remover espaços em strings também


texto = "      espaço       "
print(texto)
print(texto.lstrip())
print(texto.rstrip())
print(texto.strip())
```

Dicionários

```
carro = {
    "marca": "Toyota",
    "modelo": "Corolla",
    "ano": 2020,
    "cor": "Prata",
    "quilometragem": 45000,
    "automatico": True
}
```

Dicionários

```
# Acessando valores
print(carro["modelo"])  # Saída: Corolla

# Adicionando um novo campo
carro["preco"] = 115000

# Alterando um valor
carro["cor"] = "Preto"

# Removendo um item
del carro["quilometragem"]

# Percorrendo o dicionário
for chave, valor in carro.items():
    print(f"{chave}: {valor}")
```

Dicionários

```
# é possível isolar e percorrer individualmente
# tanto as chaves quanto os valores
print(carro.keys())
print(carro.values())

# também é possível limpar completamente o dicionário
# resultado é um dicionário vazio
carro.clear()
print(carro)
```

Sets

```
frutas = {"maçã", "banana", "laranja", "maçã"}
# 'maçã' repetida

print(frutas)
# Saída: {'maçã', 'banana', 'laranja'}  (sem
repetição)
```

Sets

```
frutas = {"maçã", "banana"}


frutas.add("uva")   # adiciona um elemento
frutas.remove("banana")   # remove um elemento


print(frutas)   # Saída: {'maçã', 'uva'}
```

Sets

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

print(a | b)    # União → {1, 2, 3, 4, 5, 6}
print(a & b)    # Interseção → {3, 4}
print(a - b)    # Diferença → {1, 2}
print(a ^ b)    # Diferença simétrica → {1, 2, 5, 6}
```

Sets

```
# é possível criar um conjunto a partir de uma lista
numeros = [1, 2, 2, 3, 3, 4]
unicos = set(numeros)

print(unicos)  # Saída: {1, 2, 3, 4}
```

Funções

```
def saudacao():
    print("Olá, mundo!")


# Chamando a função
saudacao()
```

Funções

```
def apresentar(nome):
    print(f"Prazer em te conhecer, {nome}!")


apresentar("Gabriel")
```

Funções

```
def apresentar(nome):
    print(f"Prazer em te conhecer, {nome}!")


apresentar("Gabriel")
```

Funções

```
def soma(a, b):
    return a + b


resultado = soma(5, 3)
print("Resultado:", resultado)
```

Funções

```
def soma(a, b):
    return a + b


resultado = soma(5, 3)
print("Resultado:", resultado)
```

Funções

```
def saudacao(nome="visitante"):
    print(f"Olá, {nome}!")

saudacao("Mariana")   # usa o nome passado
saudacao()            # usa o valor padrão
```

Funções

```
def calcular(a, b):
    soma = a + b
    produto = a * b
    return soma, produto


s, p = calcular(3, 4)
print(f"Soma: {s}, Produto: {p}")
```

Funções

```
def remover_duplicatas(lista):
    return list(set(lista))


nomes = ["Ana", "João", "Ana", "Carlos"]
print(remover_duplicatas(nomes))
```

Exercícios

Exercício 01

Crie um programa que recebe uma palavra do usuário e, usando um laço `for`, conta e exibe quantas vogais (a, e, i, o, u) a palavra possui. O programa não deve diferenciar maiúsculas de minúsculas.

Entrada	**Saída**
Python	A palavra 'Python' tem 1 vogal(is).

Exercício 02

Escreva um programa que peça ao usuário para inserir 5 números. Armazene esses números em uma lista e, ao final, exiba o maior valor, o menor valor e a média dos números da lista

Entrada	**Saída**
10 20 5 50 15	Valor Máximo: 50 Valor Mínimo: 5 Média: 20.0

Exercício 03

Crie um jogo onde o computador "pensa" em um número inteiro aleatório entre 1 e 20. O programa deve pedir ao usuário para adivinhar qual é o número. A cada tentativa, o programa informa se o palpite foi "Muito alto!" ou "Muito baixo!". O jogo termina quando o usuário acertar o número.

Exercício 04

Crie uma função chamada `filtrar_pares` que receba uma lista de números como argumento. A função deve retornar uma nova lista contendo apenas os números pares da lista original. Em seguida, crie um programa que peça ao usuário para inserir 5 números, chame essa função e imprima a lista de pares resultante.

Exercício 04

ENTRADAS	SAÍDAS
1 2 3 4 5	A lista de números pares é: [2, 4]
10 77 82 13 50	A lista de números pares é: [10, 82, 50]

Exercício 05

Desenvolva uma função `analisar_texto` que recebe uma frase como parâmetro. A função deve retornar um dicionário contendo três informações: a quantidade de palavras, a quantidade total de caracteres (incluindo espaços) e a palavra mais longa da frase. Crie um programa que peça uma frase ao usuário, chame a função e imprima os resultados de forma organizada.

Exercício 05

ENTRADAS	SAÍDAS
Python é uma linguagem poderosa	Quantidade de palavras: 5 Total de caracteres: 29 Palavra mais longa: poderosa
O rato roeu a roupa	Quantidade de palavras: 5 Total de caracteres: 19 Palavra mais longa: roupa

Resoluções

Resoluções podem ser encontradas em:

https://github.com/GabrielTomazini/python-saet-2025

Obrigado!