Introdução à Programação com Python

Sumário

---

1. Introdução: Bem-vindo ao Mundo da Programação!

Seja muito bem-vindo ao universo da programação! Se você está aqui, é porque tem curiosidade em transformar ideias em realidade usando um computador. Pense na programação como uma forma de conversar com o computador, dando a ele instruções claras para resolver problemas.

Python é a sua porta de entrada para esse mundo. É uma linguagem conhecida por ser muito amigável e fácil de ler, quase como se estivéssemos escrevendo em inglês. A mágica acontece quando combinamos as ferramentas certas para construir algo novo, passo a passo.

Nesta apostila, vamos mergulhar nos conceitos essenciais para construir seus primeiros programas. Prepare-se para conhecer as ferramentas que transformam ideias em realidade!

---

2. Strings: As Palavras do Computador

Uma **string** é simplesmente um texto. Em Python, qualquer coisa que esteja entre aspas simples (' ') ou aspas duplas (" ") é considerada uma string. E, assim como nós usamos palavras para formar frases, Python nos dá várias "ferramentas" para manipular esses textos.

Pense em uma variável de string como um objeto que já vem com um kit de ferramentas próprio. Para usar uma dessas ferramentas (que chamamos de **métodos**), você digita o nome da variável, seguido de um ponto (.) e o nome da ferramenta.

Métodos de Manipulação de Letras

Esses métodos são perfeitos para formatar o seu texto, seja para um título ou para comparar palavras de forma justa.

- **.upper()**: Já pensou em colocar um texto todo em letras maiúsculas para dar destaque? O método .upper() faz exatamente isso! Ele cria uma **cópia** do texto original e a devolve, só que toda em maiúsculas. A string original **não é alterada**, ela continua exatamente a mesma.
  Exemplo de uso:
  ```
  Python
  frase = "python é poderoso e versátil"

  # Criamos uma nova variável para guardar o resultado do método
  frase_maiuscula = frase.upper()
  print(frase_maiuscula)
  # Saída: PYTHON É PODEROSO E VERSÁTIL

  # Repare que a variável original 'frase' continua a mesma:
  print(frase)
  # Saída: python é poderoso e versátil
  ```

- **.lower()**: Faz o oposto do .upper(). Ele cria uma cópia do texto com todas as letras em minúsculas. Isso é super útil para comparar textos sem se preocupar com maiúsculas e minúsculas, por exemplo, em um sistema de login onde "Admin" e "admin" precisam ser a

mesma coisa.
Exemplo de uso:

```
nome = "Joana DArc"
nome_minusculo = nome.lower()
print(nome_minusculo)
# Saída: joana darc
```

- **.capitalize()**: Coloca a primeira letra da frase em maiúsculo e o restante em minúsculo. É perfeito para padronizar textos, como o início de uma frase ou de um parágrafo.
  Exemplo de uso:

```
titulo = "a programação é divertida"
titulo_formatado = titulo.capitalize()
print(titulo_formatado)
# Saída: A programação é divertida
```

Métodos de Busca e Substituição

Às vezes, você precisa encontrar ou trocar algo em um texto. Esses métodos são como um "localizar e substituir" super inteligente para a sua programação.

- **.replace(antigo, novo)**: Com este método, você pode substituir todas as ocorrências de um pedaço do texto por outro. É como pedir ao computador para trocar todas as "maçãs" por "laranjas" em uma frase.
  Exemplo de uso:

```
frase_original = "Eu gosto de maçã, mas prefiro maçã verde."
nova_frase = frase_original.replace("maçã", "banana")
print(nova_frase)
# Saída: Eu gosto de banana, mas prefiro banana verde.
```

- **.split()**: Esta é uma das ferramentas mais poderosas! Ela divide a sua string em várias partes e transforma o resultado em uma **lista** de strings. Por padrão, ele divide a string sempre que encontra um espaço. O resultado não é mais um texto único, mas sim uma coleção de palavras.
  Exemplo de uso:

```
texto = "Aprendendo a programar em Python"
palavras = texto.split()
```

```
print(palavras)
# Saída: ['Aprendendo', 'a', 'programar', 'em', 'Python']
```

- **.join()**: O método join() faz o caminho inverso do split(). Ele pega todos os itens de uma lista (que precisam ser strings) e os une em uma única string, usando o separador que você definir. É como pegar várias palavras soltas e juntá-las em uma frase.
  Exemplo de uso:
  ```
  Python
  palavras_separadas = ['Olá', 'mundo', 'da', 'programação']
  frase_unida = ' '.join(palavras_separadas)
  print(frase_unida)
  # Saída: Olá mundo da programação
  ```

O Operador in: A Magia da Busca

O operador in é uma ferramenta incrivelmente poderosa e simples. Ele não é um método, mas sim uma palavra-chave que nos permite fazer uma pergunta de "sim ou não": Essa coisa está DENTRO daquela outra coisa?

- **Para que serve?** Ele serve para verificar a existência de um item em uma sequência, seja uma substring em uma string ou um item em uma lista. O resultado é sempre um valor booleano: True (verdadeiro) se encontrar, e False (falso) se não encontrar.
- **Como usar?** A sintaxe é muito natural: item in coleção.
  Exemplo de uso:
  ```
  Python
  frase = "Eu amo programar em Python"
  palavra_chave = "Python"

  if palavra_chave in frase:
      print("A palavra 'Python' está na frase!")
  else:
      print("A palavra 'Python' não foi encontrada.")

  # Saída: A palavra 'Python' está na frase!
  ```

Funções de Contagem

- **len()**: É uma **função**, e não um método, o que significa que você a usa de forma diferente (sem o ponto). O len() serve para contar o número de itens em uma coleção. Para strings, ele conta quantos caracteres (incluindo espaços) a string tem.
  Exemplo de uso:
  ```Python
  frase = "Olá, mundo!"
  tamanho = len(frase)
  print(tamanho)
  # Saída: 11 (contando a letra 'O', a vírgula, o espaço, o 'm', etc.)
  ```

---

3. Listas: As Coleções do seu Programa

Imagine uma **lista** como uma prateleira com várias caixas, onde cada caixa pode guardar algo diferente. As listas em Python são coleções ordenadas, flexíveis e podem guardar qualquer tipo de dado, como números, textos ou até mesmo outras listas.

Acessando Itens da Lista

Para pegar um item de uma lista, usamos o **índice**, que é a posição do item. Lembre-se: em Python, a contagem dos itens sempre começa do **zero**!

- lista[índice]: Acessa o item na posição especificada. Por exemplo, frutas[0] é a "maçã".
  Exemplo de uso:
  ```Python
  numeros = [10, 20, 30, 40]
  frutas = ["maçã", "banana", "uva"]
  print(numeros[0])  # Saída: 10
  print(frutas[2])   # Saída: uva
  ```

- lista[-1]: Uma forma prática de acessar o final da lista é usando índices negativos. O índice -1 acessa o último item, -2 o penúltimo, e assim por diante.
  Exemplo de uso:
  ```Python
  numeros = [10, 20, 30, 40]
  frutas = ["maçã", "banana", "uva"]
  ```

```
print(numeros[-1]) # Saída: 40
print(frutas[-2])  # Saída: banana
```

Métodos para Modificar Listas

Esses métodos nos permitem adicionar ou remover itens, controlando o que está na nossa prateleira.

- **.append(item)**: Adiciona um novo item no **final** da lista. É uma das formas mais comuns de crescer uma lista.
  Exemplo de uso:
  ```
  Python
  lista_de_compras = ["leite", "pão"]
  lista_de_compras.append("queijo")
  print(lista_de_compras)
  # Saída: ['leite', 'pão', 'queijo']
  ```

- **.insert(indice, item)**: Adiciona um item na posição que você escolher, sem apagar nada. O restante da lista é "empurrado" para a direita.
  Exemplo de uso:
  ```
  Python
  lista = ['a', 'c', 'd']
  lista.insert(1, 'b')
  print(lista)
  # Saída: ['a', 'b', 'c', 'd']
  ```

- **.remove(item)**: Remove a **primeira ocorrência** de um item específico que você indicar.
  Exemplo de uso:
  ```
  Python
  lista_de_compras = ["leite", "pão", "queijo"]
  lista_de_compras.remove("leite")
  print(lista_de_compras)
  # Saída: ['pão', 'queijo']
  ```

- **.pop(indice)**: Remove e **retorna** o item na posição especificada. Se você não especificar um índice, ele remove e retorna o último item da lista.
  Exemplo de uso:
  ```
  Python
  lista = [10, 20, 30, 40]
  item_removido = lista.pop(2)
  ```

```
print(f"Lista após pop: {lista}")
print(f"Item removido: {item_removido}")
# Saída:
# Lista após pop: [10, 20, 40]
# Item removido: 30
```

- **.sort()**: Organiza a lista. Para números, ele ordena em ordem crescente. Para strings, ele ordena em ordem alfabética. Este método altera a lista original.
  Exemplo de uso:
  Python
  ```
  numeros = [4, 1, 3, 2]
  numeros.sort()
  print(numeros)
  # Saída: [1, 2, 3, 4]
  ```

- **.reverse()**: Inverte a ordem dos elementos da lista, sem ordená-los. Este método altera a lista original.
  Exemplo de uso:
  Python
  ```
  lista = ['a', 'b', 'c']
  lista.reverse()
  print(lista)
  # Saída: ['c', 'b', 'a']
  ```

- **.index(item)**: Retorna o índice (a posição) da **primeira ocorrência** do item especificado. Se o item não for encontrado, ele causa um erro.
  Exemplo de uso:
  Python
  ```
  lista = ['pão', 'queijo', 'leite', 'pão']
  posicao_pao = lista.index('pão')
  print(posicao_pao)
  # Saída: 0
  ```

---

4. Repetição: O Poder do for e range()

O loop **for** é uma das estruturas mais poderosas em programação. Ele é ideal para percorrer os itens de uma lista, um por um, e executar um bloco de código para cada item.

Loop for em listas

Imagine que você tem uma lista de frutas e quer imprimir cada uma. Em vez de escrever um print() para cada fruta, o for faz isso de forma elegante.

Exemplo de uso:
```python
frutas = ["maçã", "banana", "uva"]

# Para cada 'fruta' dentro da lista 'frutas', faça o seguinte:
for fruta in frutas:
    print(fruta)
# Saída:
# maçã
# banana
# uva
```

A Função range()

A função **range()** é uma forma inteligente de criar uma sequência de números, o que é perfeito para usar com o for quando você precisa repetir algo um número exato de vezes.

- range(5): Cria uma sequência de 0 até 4. O número final (5) **não é incluído**.
- range(1, 6): Cria uma sequência de 1 até 5.
- range(0, 10, 2): Cria uma sequência de 0 até 9, mas pula de 2 em 2 (0, 2, 4, 6, 8).
  Exemplo de uso:
  ```Python
  for numero in range(1, 6):
      print(numero)
  # Saída:
  # 1
  # 2
  # 3
  # 4
  # 5
  ```

Contagem Reversa

E se você precisar percorrer a lista de trás para frente? O range() também faz isso! Basta dizer o ponto de partida, o de chegada e o passo.

Exemplo de uso:
```python
numeros = [10, 20, 30, 40, 50]
# Para contar para trás, o range() precisa de 3 valores:
# range(início, parada, passo)
# Início: len(numeros) - 1, que é o último índice (4).
# Parada: -1 (o loop vai até o índice 0, parando antes de chegar no -1).
# Passo: -1 (para contar para trás, de um em um).
for i in range(len(numeros) - 1, -1, -1):
    print(numeros[i])
# Saída:
# 50
# 40
# 30
# 20
# 10
```

---

5. Aleatoriedade: O Módulo random

Em muitos programas, precisamos de algo aleatório, como um jogo de dados ou um sorteio. Para isso, Python nos oferece o módulo **random**. Um **módulo** é como uma "maleta de ferramentas" extra que podemos usar. Para usá-la, você precisa "abrí-la" com o comando import no início do seu código.

- import random: Importa o módulo, tornando suas ferramentas disponíveis.

- random.randint(a, b): Retorna um número inteiro aleatório entre a e b.
- random.choice(lista): Escolhe um item aleatoriamente de uma lista.
  Exemplo de uso:
  Python
  ```
  import random

  # Sorteia um número entre 1 e 10
  numero_secreto = random.randint(1, 10)
  print(f"Número secreto gerado: {numero_secreto}")

  # Sorteia um jogador de uma lista
  jogadores = ["Maria", "João", "Ana", "Pedro"]
  sorteado = random.choice(jogadores)
  print(f"O sorteado foi: {sorteado}")
  ```

Jogo Prático: Pedra, Papel e Tesoura!

Este é um exemplo clássico que combina o módulo random, listas e o loop while para criar uma experiência interativa. A lógica é simples: o computador faz uma escolha aleatória, você faz a sua, e o programa decide quem vence.

Python

```
import random

# A lista de opções do jogo
opcoes = ["pedra", "papel", "tesoura"]

# Usamos um loop 'while True' para que o jogo continue rodando
# até que o usuário decida parar.
while True:
    # A escolha do computador é aleatória
    escolha_computador = random.choice(opcoes)

    # A escolha do usuário. Usamos o .lower() para garantir que a entrada seja minúscula
    escolha_usuario = input("Escolha Pedra, Papel ou Tesoura: ").lower()
```

```
    # Verificamos se a escolha do usuário é válida
    if escolha_usuario not in opcoes:
        print("Opção inválida. Tente novamente.")
        continue # O 'continue' volta para o início do loop

    print(f"O computador escolheu: {escolha_computador}")

    # A lógica do jogo: usando estruturas 'if' e 'elif'
    if escolha_usuario == escolha_computador:
        print("Empate!")
    elif escolha_usuario == "pedra" and escolha_computador == "tesoura":
        print("Você Venceu!")
    elif escolha_usuario == "papel" and escolha_computador == "pedra":
        print("Você Venceu!")
    elif escolha_usuario == "tesoura" and escolha_computador == "papel":
        print("Você Venceu!")
    else:
        print("Você Perdeu!")

    # Pergunta se o usuário quer jogar novamente
    jogar_novamente = input("Jogar novamente? (s/n): ").lower()
    if jogar_novamente != "s":
        break # O 'break' sai do loop, encerrando o jogo

print("Obrigado por jogar!")
```

---

6. Exercícios Práticos em Etapas

Agora é sua vez de praticar! Use o que você aprendeu para resolver os problemas abaixo. Siga as etapas para guiar seu raciocínio, e o gabarito estará na próxima seção.

Exercício 1: Exibindo Itens de uma Lista

1. Crie uma variável chamada frutas e atribua a ela uma lista com 3 nomes de frutas (strings).

2. Use um loop **for** para percorrer cada item da lista frutas.
3. Dentro do loop, use o **print()** para exibir o nome de cada fruta em uma nova linha.

Exercício 2: Contando Caracteres de uma Palavra

1. Use o **input()** para pedir ao usuário que digite uma palavra e guarde-a em uma variável chamada palavra.
2. Use a função **len()** para contar o número de caracteres da palavra. Guarde o resultado em uma variável chamada tamanho.
3. Use o **print()** para exibir uma mensagem informando o tamanho da palavra.

Exercício 3: Criando uma Lista com o Usuário

1. Crie uma lista vazia chamada amigos.
2. Crie uma variável chamada contador e defina seu valor inicial como 0.
3. Use um loop **while** que continue rodando **enquanto** o contador for menor que 3.
4. Dentro do loop, use o **input()** para pedir o nome de um amigo e guarde-o em uma variável.
5. Use o método **.append()** para adicionar o nome do amigo à lista amigos.
6. Aumente o contador em 1 a cada volta do loop.
7. Após o loop, use o **print()** para exibir a lista completa.

Exercício 4: Imprimindo uma Sequência de Números

1. Use um loop **for** e a função **range()** para gerar os números de 1 até 5.
2. Dentro do loop, use o **print()** para exibir cada número.

Exercício 5: Números Pares com range()

1. Use um loop **for** e a função **range()** para gerar os números pares de 2 até 10.
2. Dentro do loop, use o **print()** para exibir cada número.

Exercício 6: Imprimindo uma Lista de Trás para Frente

1. Crie uma lista com 5 números inteiros.
2. Use um loop **for** e a função **range()** para percorrer os índices da lista de trás para frente.
3. Dentro do loop, use o **print()** para exibir o número correspondente a cada índice.

Exercício 7: Adivinhe o Número

1. Importe o módulo **random**.
2. Use a função **random.randint()** para sortear um número entre 1 e 10 e guarde-o em uma variável.
3. Peça ao usuário para adivinhar o número usando o **input()** e guarde o palpite em uma variável (lembre-se de converter para número com **int()**).
4. Use uma estrutura **if** para verificar se o palpite do usuário é igual ao número sorteado.
5. Exiba uma mensagem de "Parabéns!" se ele acertar, ou "Que pena!" se errar, informando qual era o número correto.

Exercício 8: Buscando uma Palavra em uma Frase

1. Peça ao usuário que digite uma frase e guarde-a em uma variável.
2. Peça ao usuário que digite uma palavra e guarde-a em outra variável.
3. Use o operador **in** dentro de uma estrutura **if** para verificar se a palavra está contida na frase.
4. Exiba uma mensagem informando se a palavra foi encontrada ou não.

Exercício 9: Somando Elementos de uma Lista

1. Crie uma lista de números.
2. Crie uma variável chamada soma e defina seu valor inicial como 0.
3. Use um loop **for** para percorrer cada número na lista.
4. Dentro do loop, adicione o valor do número atual à variável soma.
5. Após o loop, exiba o valor final da soma.

Exercício 10: Validando uma Senha Simples

1. Crie duas variáveis, usuario_original e senha_original, com valores pré-definidos.
2. Peça ao usuário para digitar um nome de usuário e uma senha.
3. Use uma estrutura **if** com a condição e (**and**) para verificar se o nome de usuário E a senha estão corretos.
4. Exiba uma mensagem de sucesso ou erro.

Exercício 11: Capitalizando Nomes em uma Lista

1. Crie uma lista com 3 nomes, todos em letras minúsculas (ex: ['joão', 'maria', 'pedro']).
2. Crie uma nova lista vazia chamada nomes_formatados.
3. Use um loop **for** para percorrer a primeira lista.
4. Dentro do loop, use o método **.capitalize()** em cada nome.
5. Use o método **.append()** para adicionar o nome formatado à nova lista.
6. Ao final, imprima a lista nomes_formatados.

Exercício 12: Sistema de Login com Tentativas

1. Defina um nome de usuário e uma senha corretos em variáveis.
2. Crie uma variável tentativas com o valor inicial de 3.
3. Use um loop **while** que continue enquanto o número de tentativas for maior que 0.
4. Dentro do loop, peça o nome de usuário e a senha.
5. Use uma estrutura **if** para verificar se o login está correto. Se sim, imprima "Login bem-sucedido!" e use o comando **break** para sair do loop.
6. Se o login estiver incorreto, imprima "Usuário ou senha incorretos." e diminua a variável tentativas em 1.
7. Após o loop, use o **if** para verificar se as tentativas chegaram a 0. Se sim, imprima "Você excedeu o número de tentativas.".

Exercício 13: Verificador de Palíndromo

1. Peça ao usuário para digitar uma palavra.
2. Use o método **.lower()** para converter a palavra para minúsculas, garantindo que "Arara" e "arara" sejam tratadas da mesma forma.
3. Crie uma nova variável para guardar a palavra reversa. Uma forma simples é usar fatiamento de string: palavra[::-1].
4. Use uma estrutura **if** para comparar a palavra original com a palavra reversa.
5. Exiba se a palavra é ou não um palíndromo (uma palavra que se lê da mesma forma de trás para frente).

Exercício 14: Contando Ocorrências de uma Letra

1. Peça ao usuário para digitar uma frase.
2. Peça ao usuário para digitar uma letra.
3. Crie uma variável contador_letras com o valor 0.
4. Use um loop **for** para percorrer cada caractere da frase.
5. Dentro do loop, use um **if** para verificar se o caractere atual é igual à letra que o usuário digitou. Use o método **.lower()** para ignorar maiúsculas e minúsculas.
6. Se for, adicione 1 à variável contador_letras.
7. Ao final do loop, imprima quantas vezes a letra apareceu na frase.

Exercício 15: Sorteando um Aluno para Apresentação

1. Crie uma lista com pelo menos 5 nomes de alunos.

2. Importe o módulo **random**.
3. Use a função **random.choice()** para sortear um nome da lista.
4. Exiba uma mensagem dizendo "O aluno sorteado para a apresentação é:" seguido do nome do aluno.

---

7. Gabarito

Exercício 1

Python

```
frutas = ["maçã", "banana", "uva"]
for fruta in frutas:
    print(fruta)
```

Exercício 2

Python

```
palavra = input("Digite uma palavra: ")
tamanho = len(palavra)
print(f"A palavra '{palavra}' tem {tamanho} letras.")
```

Exercício 3

Python

```
amigos = []
contador = 0
while contador < 3:
```

```
    nome = input("Digite o nome de um amigo: ")
    amigos.append(nome)
    contador = contador + 1
print("Lista de amigos:", amigos)
```

Exercício 4

```
    Python
```

```
for numero in range(1, 6):
    print(numero)
```

Exercício 5

```
    Python
```

```
for numero in range(2, 11, 2):
    print(numero)
```

Exercício 6

```
    Python
```

```
numeros = [10, 20, 30, 40, 50]
for i in range(len(numeros) - 1, -1, -1):
    print(numeros[i])
```

Exercício 7

```
    Python
```

```python
import random

numero_secreto = random.randint(1, 10)
palpite = int(input("Adivinhe o número (entre 1 e 10): "))

if palpite == numero_secreto:
    print("Parabéns! Você acertou!")
else:
    print(f"Que pena! Você errou. O número era {numero_secreto}.")
```

Exercício 8

Python

```python
frase = input("Digite uma frase: ")
palavra = input("Digite uma palavra: ")

if palavra in frase:
    print(f"A palavra '{palavra}' está na frase.")
else:
    print(f"A palavra '{palavra}' NÃO está na frase.")
```

Exercício 9

Python

```python
lista_numeros = [10, 20, 30, 40]
soma = 0
for numero in lista_numeros:
    soma = soma + numero
print("A soma dos números é:", soma)
```

Exercício 10

Python

```
usuario_original = "admin"
senha_original = "senha123"

usuario_digitado = input("Digite o nome de usuário: ")
senha_digitada = input("Digite a senha: ")

if usuario_digitado == usuario_original and senha_digitada == senha_original:
    print("Login bem-sucedido!")
else:
    print("Usuário ou senha incorretos.")
```

Exercício 11

Python

```
nomes = ['joão', 'maria', 'pedro']
nomes_formatados = []

for nome in nomes:
    nomes_formatados.append(nome.capitalize())

print(nomes_formatados)
```

Exercício 12

Python

```
usuario_correto = "admin"
senha_correta = "12345"
tentativas = 3

while tentativas > 0:
    usuario_digitado = input("Usuário: ")
```

```
    senha_digitada = input("Senha: ")

    if usuario_digitado == usuario_correto and senha_digitada == senha_correta:
        print("Login bem-sucedido!")
        break
    else:
        tentativas = tentativas - 1
        print(f"Usuário ou senha incorretos. Tentativas restantes: {tentativas}")

if tentativas == 0:
    print("Você excedeu o número de tentativas.")
```

Exercício 13

Python

```
palavra = input("Digite uma palavra: ").lower()
palavra_reversa = palavra[::-1]

if palavra == palavra_reversa:
    print(f"A palavra '{palavra}' é um palíndromo!")
else:
    print(f"A palavra '{palavra}' não é um palíndromo.")
```

Exercício 14

Python

```
frase = input("Digite uma frase: ")
letra = input("Digite uma letra para contar: ")
contador_letras = 0

for caractere in frase:
    if caractere.lower() == letra.lower():
        contador_letras = contador_letras + 1

print(f"A letra '{letra}' apareceu {contador_letras} vezes na frase.")
```

Exercício 15

Python

```
import random

alunos = ["Ana", "Pedro", "Maria", "João", "Clara"]
aluno_sorteado = random.choice(alunos)

print("O aluno sorteado para a apresentação é:")
print(aluno_sorteado)
```