

Material de Apoio Python

Introdução

👋 Bem-vindos ao material de apoio Python! Aqui, vamos aprender de forma divertida e didática os conceitos básicos de programação, como variáveis, estruturas de decisão, estruturas de repetição, funções, listas, tuplas, dicionários e a programação orientada a objetos. Vamos começar?

1. 📦 Variáveis

O que são Variáveis?


🤖 **Variáveis** são como caixinhas onde podemos guardar informações. Essas informações podem ser números, textos, etc. Em Python, não precisamos declarar o tipo da variável, basta atribuir um valor a ela.

Tipos de Variáveis:

- **Inteiros (int):** números sem casas decimais.
- **Ponto flutuante (float):** números com casas decimais.
- **Strings (str):** textos.
- **Booleanos (bool):** valores verdadeiros ou falsos.

🧠 Exemplos:

python

 Copiar código

```
idade = 25 # inteiro
nome = "Ana" # string
altura = 1.70 # ponto flutuante
estudante = True # booleano
```

📄 Exercício 1:

1. Crie uma variável chamada `cidade` e atribua a ela o nome da cidade onde você mora.
2. Crie uma variável chamada `ano` e atribua a ela o ano atual.
3. Crie uma variável chamada `temperatura` e atribua a ela a temperatura atual.

📺 Dica visual:

Imagine que variáveis são caixinhas etiquetadas, cada uma com um valor dentro!

2. 📁 Estruturas de Decisão

O que são Estruturas de Decisão?

🤖 **Estruturas de decisão** permitem que o programa tome decisões baseadas em condições. Em Python, usamos as palavras-chave `if`, `elif` e `else`.

Sintaxe:

```
python Copiar código

if condição:
    # bloco de código executado se a condição for verdadeira
elif outra_condição:
    # bloco de código executado se a outra_condição for verdadeira
else:
    # bloco de código executado se nenhuma condição for verdadeira
```

🧠 Exemplos:

```
python Copiar código

idade = 18
if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

📄 Exercício 2:


4. Crie uma variável chamada `nota` e atribua a ela um valor entre 0 e 10.
5. Se a `nota` for maior ou igual a 6, imprima "Aprovado".
6. Se a `nota` for menor que 6, imprima "Reprovado".

📺 Dica visual:

Imagine uma bifurcação na estrada: se a condição for verdadeira, siga por um caminho; caso contrário, siga pelo outro.

3. Estruturas de Repetição

O que são Estruturas de Repetição?

 **Estruturas de repetição** permitem que uma ação seja repetida várias vezes. Em Python, usamos `for` e `while`.

Sintaxe do `for`:

```
python                                                                    Copiar código

for variável in sequência:
    # bloco de código a ser repetido
```

Sintaxe do `while`:

```
python                                                                    Copiar código

while condição:
    # bloco de código a ser repetido enquanto a condição for verdadeira
```

Exemplos:

```
python                                                                    Copiar código

# Usando for
for i in range(5):
    print(i)

# Usando while
contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

Exercício 3:

7. Use um loop `for` para imprimir os números de 1 a 10.
8. Use um loop `while` para imprimir os números de 10 a 1.

Dica visual:

Pense em uma roda-gigante: cada volta é uma repetição!

4. 📞 Funções

O que são Funções?

🔧 **Funções** são blocos de código que realizam uma tarefa específica e podem ser reutilizados. Usamos a palavra-chave `def` para definir uma função.

Sintaxe de uma Função:

python

📄 Copiar código

```
def nome_da_funcao(parametros):  
    # bloco de código  
    return valor_de_retorno
```

Funções com e sem parâmetros:

- Sem parâmetros:

python

📄 Copiar código

```
def saudacao():  
    print("Olá, seja bem-vindo!")  
  
saudacao()
```

- Com parâmetros:

python

📄 Copiar código

```
def saudacao(nome):  
    print(f"Olá, {nome}, seja bem-vindo!")  
  
saudacao("Ana")
```

Funções com retorno:


python

📄 Copiar código

```
def soma(a, b):  
    return a + b  
  
resultado = soma(5, 3)  
print(resultado) # 8
```

Exemplos:

python

 Copiar código

```
def multiplicar(a, b):  
    return a * b  
  
resultado = multiplicar(4, 5)  
print(resultado) # 20
```

Exercício 4:


9. Crie uma função chamada soma que recebe dois números como parâmetros e retorna a soma deles.
10. Chame a função soma com os valores 5 e 3 e imprima o resultado.
11. Crie uma função chamada subtrair que recebe dois números como parâmetros e retorna a subtração do segundo número pelo primeiro.
12. Chame a função subtrair com os valores 10 e 4 e imprima o resultado.

Dica visual:

Imagine que funções são como telefones: você liga para eles para realizar uma tarefa específica!


5. Listas

O que são Listas?

 **Listas** são coleções ordenadas de itens. Podemos adicionar, remover e modificar itens em uma lista.

Sintaxe:

python


 Copiar código

```
lista = [item1, item2, item3]
```

Métodos importantes:

- Adicionar item (append):


python

 Copiar código

```
frutas = ["maçã", "banana"]
frutas.append("laranja")
print(frutas) # ["maçã", "banana", "laranja"]
```

- Remover item (remove):


python

 Copiar código

```
frutas.remove("banana")
print(frutas) # ["maçã", "laranja"]
```

Exemplos:

python

 Copiar código

```
frutas = ["maçã", "banana", "laranja"]
print(frutas[0]) # maçã

frutas.append("uva")
print(frutas) # ["maçã", "banana", "laranja", "uva"]

frutas.remove("banana")
print(frutas) # ["maçã", "laranja", "uva"]
```

Exercício 5:


13. Crie uma lista chamada `animais` contendo três animais de sua escolha.
14. Adicione mais um animal à lista `animais` usando o método `append`.
15. Remova o primeiro animal da lista `animais` usando o método `remove`.

Dica visual:

Pense em uma lista de compras: cada item é um elemento na lista!


6. Tuplas

O que são Tuplas?

 **Tuplas** são coleções ordenadas de itens, imutáveis (não podemos modificar os itens depois de criadas).

Sintaxe:


python

 Copiar código

```
tupla = (item1, item2, item3)
```

Exemplos:

python

 Copiar código

```
cores = ("vermelho", "verde", "azul")
print(cores[1]) # verde

# Tentando modificar uma tupla resultará em erro
# cores[1] = "amarelo" # TypeError
```

Exercício 6:


16. Crie uma tupla chamada meses contendo os 12 meses do ano.
17. Acesse e imprima o quarto mês da tupla meses.

Dica visual:

Imagine que tuplas são como prateleiras fixas: você não pode mudar a ordem dos itens depois de colocar.


7. Dicionários

O que são Dicionários?

 **Dicionários** são coleções de pares chave-valor. Usamos chaves para acessar os valores associados.

Sintaxe:


python

 Copiar código

```
dicionario = {"chave1": valor1, "chave2": valor2}
```

Exemplos:

python

 Copiar código

```
aluno = {"nome": "Ana", "idade": 20, "curso": "Engenharia"}
print(aluno["nome"]) # Ana

aluno["idade"] = 21
print(aluno) # {"nome": "Ana", "idade": 21, "curso": "Engenharia"}

del aluno["curso"]
print(aluno) # {"nome": "Ana", "idade": 21}
```

Exercício 7:


18. Crie um dicionário chamado `livro` com as chaves `título`, `autor` e `ano`.
19. Acesse e imprima o valor da chave `autor`.
20. Adicione uma nova chave `editora` ao dicionário `livro`.

Dica visual:

Pense em um dicionário real: cada palavra (chave) tem uma definição (valor).

8. Programação Orientada a Objetos (POO)


O que é POO?

 **POO** é um paradigma de programação que usa "objetos" para modelar dados e comportamentos. Em Python, usamos a palavra-chave `class` para definir uma classe.

Classe:


 Uma **classe** é como um molde para criar objetos. Ela define atributos e métodos.

python

 Copiar código

```
class Carro:
    pass
```


Atributos:

 **Atributos** são as características dos objetos criados a partir de uma classe.


```
python Copiar código

class Carro:
    def __init__(self, marca, ano):
        self.marca = marca
```


Atributo Privado:

 **Atributo privado** é um atributo que só pode ser acessado dentro da classe. Usamos `__` (dois underscores) antes do nome do atributo.

```
python Copiar código

class Carro:
    def __init__(self, marca, ano):
        self.__marca = marca
        self.ano = ano
```

Métodos:


 **Métodos** são funções definidas dentro de uma classe.

```
python Copiar código

class Carro:
    def __init__(self, marca, ano):
        self.marca = marca
        self.ano = ano

    def descrever(self):
        print(f"Este carro é da marca {self.marca} e foi fabricado no ano {self.ano}.")
```


Método Construtor:

 **Método construtor** é um método especial chamado `__init__`, que é chamado automaticamente quando um objeto da classe é criado.


```
python Copiar código

class Carro:
    def __init__(self, marca, ano):
        self.marca = marca
        self.ano = ano
```

Métodos get e set:

 **Métodos get e set** são usados para obter e modificar os valores dos atributos privados.

python

 Copiar código


```
class Carro:
    def __init__(self, marca, ano):
        self.__marca = marca
        self.ano = ano

    def get_marca(self):
        return self.__marca

    def set_marca(self, marca):
        self.__marca = marca
```

Exemplos:

python

 Copiar código

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def apresentar(self):
        print(f"Meu nome é {self.nome} e eu tenho {self.idade} anos.")

pessoa1 = Pessoa("Carlos", 30)
pessoa1.apresentar()
```

Exercício 8:

21. Crie uma classe chamada Carro com os atributos marca e ano.
22. Adicione um método chamado descrever que imprime "Este carro é da marca X e foi fabricado no ano Y", onde X e Y são os valores dos atributos marca e ano.
23. Crie um objeto da classe Carro e chame o método descrever.

Dica visual:

Imagine que a POO é como construir uma casa: você cria um projeto (classe) e depois constrói a casa (objeto) com base nesse projeto.