Fundamentos de Python

Sumário

1. Variáveis e Tipos de Dados

Definição

Uma **variável** é um nome associado a um valor armazenado na memória do computador. Em Python, o tipo de uma variável é determinado automaticamente no momento da atribuição.

Os **tipos de dados** indicam qual é a natureza do valor armazenado e quais operações fazem sentido para ele.

Tipos de Dados Primitivos

`int` (Inteiro)

Representa números inteiros, positivos ou negativos, sem parte decimal.

```
idade = 25
temperatura = -10
ano = 2024
```

`float` (Ponto Flutuante)

Representa números reais com parte decimal.

```
altura = 1.75
pi = 3.14159
temperatura = -5.5
```

`bool` (Booleano)

Representa valores lógicos: `True` (verdadeiro) ou `False` (falso).

```
esta_chovendo = True
maior_de_idade = False
```

`str` (String)

Representa sequências de caracteres (texto). Pode ser delimitada por aspas simples ou duplas.

```
nome = "Maria"
sobrenome = 'Silva'
frase = "Python é uma linguagem poderosa"
```

Exemplos

```
# Declaração e atribuição de variáveis
nome_completo = "João Pedro Santos"
idade = 28
altura = 1.82
estudante = True

# Verificando o tipo de uma variável
print(type(nome_completo))  # <class 'str'>
print(type(idade))          # <class 'int'>
print(type(altura))         # <class 'float'>
print(type(estudante))      # <class 'bool'>

# Conversão entre tipos (casting)
numero_texto = "42"
numero_inteiro = int(numero_texto)
print(numero_inteiro + 8)  # 50

preco = 19.99
preco_inteiro = int(preco)  # 19 (trunca a parte decimal)

# Concatenação de strings
primeiro_nome = "Ana"
ultimo_nome = "Costa"
nome_completo = primeiro_nome + " " + ultimo_nome
print(nome_completo)  # Ana Costa
```

Exercícios

1. Crie variáveis para armazenar seu nome, idade, altura e se você é estudante. Imprima cada variável com seu tipo.
2. Declare uma variável com valor `"123"` (string) e converta-a para inteiro. Some 50 a esse valor e imprima o resultado.
3. Crie duas variáveis do tipo float representando a base e altura de um retângulo. Calcule e imprima a área.

4. Declare uma variável booleana chamada `tem_carteira` e atribua um valor. Imprima o tipo e o valor dessa variável.

Desafios

1. Crie um programa que declare variáveis para armazenar o preço de três produtos (float). Calcule o total e converta o resultado para inteiro (truncando os centavos). Imprima a diferença entre o total real e o total truncado.
2. Declare uma string contendo um número decimal (ex: "3.14") e converta-a primeiro para float e depois para int. Explique o que acontece em cada conversão.

Exemplos de uso

- **Cadastros simples**: armazenamento de nome, idade e outros dados básicos
- **Cálculos numéricos**: operações com valores decimais
- **Regras condicionais**: decisões baseadas em verdadeiro ou falso
- **Texto**: leitura, junção e transformação de strings

---

2. Entrada e Saída Básica

Definição

Entrada é a leitura de dados fornecidos pelo usuário. **Saída** é a exibição de informações na tela.

Em Python:
- `input()`: função para capturar entrada do usuário (sempre retorna string)
- `print()`: função para exibir saída na tela

Exemplos

```
# Saída simples
print("Olá, mundo!")
print("Python", "é", "incrível")  # Múltiplos argumentos

# Saída formatada
nome = "Carlos"
idade = 30
print(f"Meu nome é {nome} e tenho {idade} anos")  # f-string
print("Meu nome é {} e tenho {} anos".format(nome, idade))  # format()
print("Meu nome é", nome, "e tenho", idade, "anos")  # vírgulas

# Entrada básica
nome_usuario = input("Digite seu nome: ")
print(f"Bem-vindo, {nome_usuario}!")

# Entrada com conversão de tipo
idade_str = input("Digite sua idade: ")
idade_int = int(idade_str)
print(f"Ano que vem você terá {idade_int + 1} anos")

# Forma compacta
idade = int(input("Digite sua idade: "))
```

Parâmetros comuns do `print()`

```
# sep: define separador entre argumentos
print("Python", "Java", "C++", sep=" | ")  # Python | Java | C++

# end: define o que aparece no final (padrão é '\n')
print("Carregando", end="...")
print("Concluído!")  # Aparece na mesma linha

# Imprimindo múltiplas linhas
print("Linha 1\nLinha 2\nLinha 3")
```

Exercícios

1. Escreva um programa que peça o nome do usuário e imprima uma saudação personalizada.
2. Crie um programa que peça dois números ao usuário e imprima a soma deles.
3. Faça um programa que pergunte ao usuário seu nome, idade e cidade, depois imprima uma frase usando todos esses dados com f-string.
4. Escreva um programa que leia um número decimal do usuário e imprima esse número com apenas duas casas decimais.

Desafios

1. Crie um programa que peça ao usuário três notas (podem ser decimais) e calcule a média. Imprima a média formatada com duas casas decimais e uma mensagem indicando se o aluno foi aprovado (média >= 7).
2. Desenvolva um conversor de temperaturas que leia a temperatura em Celsius e imprima o equivalente em Fahrenheit e Kelvin. Use a fórmula: F = C × 9/5 + 32 e K = C + 273.15.

Exemplos de uso

- **Programas interativos**: entrada e resposta em tempo real
- **Scripts de terminal**: automação de tarefas
- **Cálculos simples**: processamento direto de valores digitados
- **Back-end**: leitura de dados enviados por formulários

---

3. Operadores

Definição

Operadores são símbolos especiais usados para calcular, comparar ou combinar valores.

3.1 Operadores Aritméticos

Realizam operações matemáticas básicas.

Operador	Operação	Exemplo	Resultado
`+`	Adição	`5 + 3`	`8`
`-`	Subtração	`5 - 3`	`2`
`*`	Multiplicação	`5 * 3`	`15`
`/`	Divisão (float)	`5 / 2`	`2.5`
`//`	Divisão inteira	`5 // 2`	`2`
`%`	Módulo (resto)	`5 % 2`	`1`
`**`	Exponenciação	`5 ** 2`	`25`

```
# Exemplos
a = 10
b = 3

soma = a + b          # 13
diferenca = a - b     # 7
produto = a * b       # 30
divisao = a / b       # 3.333...
divisao_inteira = a // b  # 3
resto = a % b         # 1
potencia = a ** b     # 1000

# Operador unário
negativo = -a         # -10
```

3.2 Operadores de Comparação

Comparam dois valores e retornam `True` ou `False`.

Operador	Significado	Exemplo	Resultado
`==`	Igual a	`5 == 5`	`True`
`!=`	Diferente de	`5 != 3`	`True`
`>`	Maior que	`5 > 3`	`True`
`<`	Menor que	`5 < 3`	`False`
`>=`	Maior ou igual	`5 >= 5`	`True`
`<=`	Menor ou igual	`5 <= 3`	`False`

```
x = 10
y = 20

print(x == y)   # False
print(x != y)   # True
print(x < y)    # True
print(x > y)    # False
print(x <= 10)  # True
print(y >= 20)  # True
```

3.3 Operadores Lógicos

Combinam expressões booleanas.

Operador	Descrição	Exemplo	Resultado
`and`	E lógico (ambos verdadeiros)	`True and False`	`False`
`or`	Ou lógico (pelo menos um verdadeiro)	`True or False`	`True`
`not`	Negação lógica	`not True`	`False`

```
idade = 25
tem_carteira = True

# Pode dirigir se tem 18+ anos E tem carteira
pode_dirigir = idade >= 18 and tem_carteira  # True

# Desconto se é estudante OU idoso
estudante = False
idoso = True
tem_desconto = estudante or idoso  # True

# Negação
nao_tem_carteira = not tem_carteira  # False
```

Tabela Verdade:

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

3.4 Operadores de Atribuição

Atribuem valores a variáveis.

Operador	Exemplo	Equivalente a
`=`	`x = 5`	`x = 5`
`+=`	`x += 3`	`x = x + 3`
`-=`	`x -= 3`	`x = x - 3`
`*=`	`x *= 3`	`x = x * 3`
`/=`	`x /= 3`	`x = x / 3`
`//=`	`x //= 3`	`x = x // 3`
`%=`	`x %= 3`	`x = x % 3`
`**=`	`x **= 3`	`x = x ** 3`

```
contador = 0
contador += 1  # contador agora é 1
contador += 1  # contador agora é 2

preco = 100
preco *= 1.1  # aumenta 10%, preco agora é 110.0
```

Exemplos Práticos

```
# Calculando área de um círculo
raio = 5
pi = 3.14159
area = pi * raio ** 2
print(f"Área do círculo: {area:.2f}")

# Verificando se um número é par
numero = 42
eh_par = numero % 2 == 0
print(f"{numero} é par? {eh_par}")

# Verificando elegibilidade para votar
idade = 17
pode_votar = idade >= 16
voto_obrigatorio = idade >= 18 and idade < 70
print(f"Pode votar: {pode_votar}")
print(f"Voto obrigatório: {voto_obrigatorio}")

# Calculando desconto
preco_original = 100
tem_cupom = True
eh_cliente_vip = False

desconto = (tem_cupom or eh_cliente_vip) and preco_original > 50
if desconto:
    preco_final = preco_original * 0.9
else:
    preco_final = preco_original
```

Exercícios

1. Escreva um programa que leia dois números e imprima a soma, subtração, multiplicação e divisão entre eles.
2. Crie um programa que calcule o resto da divisão de um número por outro (ambos fornecidos pelo usuário).

3. Verifique se um número fornecido pelo usuário é divisível por 3 e por 5 simultaneamente.
4. Leia a idade de uma pessoa e verifique se ela é maior de idade (>= 18 anos).

Desafios

1. Crie um programa que calcule o IMC (Índice de Massa Corporal) de uma pessoa. Fórmula: IMC = peso / altura². O programa deve verificar se o IMC está na faixa saudável (18.5 <= IMC < 25).
2. Desenvolva um programa que leia três números e determine se eles podem formar um triângulo (cada lado deve ser menor que a soma dos outros dois) e, se sim, se é equilátero (todos os lados iguais), isósceles (dois lados iguais) ou escaleno (todos os lados diferentes).

Aplicações

- **Cálculos financeiros**: juros, descontos, parcelas
- **Sistemas de validação**: verificar elegibilidade, permissões
- **Jogos**: calcular pontuação, verificar colisões
- **Análise de dados**: comparações estatísticas

---

4. Controle de Fluxo: `if / elif / else`

Definição

Estruturas de controle de fluxo permitem que o programa tome decisões e execute diferentes blocos de código baseados em condições.

Sintaxe básica:

```
if condicao:
    # código executado se condição for True
elif outra_condicao:
    # código executado se primeira condição for False e esta for True
else:
    # código executado se todas as condições anteriores forem False
```

Características Importantes

- A indentação (4 espaços ou 1 tab) é obrigatória e define o bloco de código
- `elif` é abreviação de "else if"
- `else` é opcional
- Pode haver múltiplos `elif`
- Apenas o primeiro bloco verdadeiro é executado

Exemplos

```
# if simples
idade = 20
if idade >= 18:
    print("Você é maior de idade")

# if-else
temperatura = 15
if temperatura > 25:
    print("Está quente")
else:
    print("Está frio ou agradável")

# if-elif-else
nota = 85
if nota >= 90:
    conceito = "A"
elif nota >= 80:
    conceito = "B"
elif nota >= 70:
    conceito = "C"
elif nota >= 60:
    conceito = "D"
else:
    conceito = "F"
print(f"Seu conceito é: {conceito}")

# Condições compostas
idade = 25
tem_carteira = True
if idade >= 18 and tem_carteira:
    print("Você pode dirigir")
else:
    print("Você não pode dirigir")

# if aninhado
idade = 17
tem_autorizacao = True
if idade >= 18:
    print("Entrada permitida")
```

```
else:
    if tem_autorizacao:
        print("Entrada permitida com autorização")
    else:
        print("Entrada negada")

# Operador ternário (condicional em linha)
idade = 20
status = "maior" if idade >= 18 else "menor"
print(f"Você é {status} de idade")
```

Exemplos Práticos

```
# Sistema de login simples
usuario_correto = "admin"
senha_correta = "1234"

usuario = input("Usuário: ")
senha = input("Senha: ")

if usuario == usuario_correto and senha == senha_correta:
    print("Login realizado com sucesso!")
elif usuario == usuario_correto:
    print("Senha incorreta")
else:
    print("Usuário não encontrado")

# Calculadora de preço com descontos
preco = float(input("Preço do produto: R$ "))
quantidade = int(input("Quantidade: "))

total = preco * quantidade

if quantidade >= 10:
    desconto = 0.15
elif quantidade >= 5:
    desconto = 0.10
else:
    desconto = 0

total_final = total * (1 - desconto)
print(f"Total: R$ {total:.2f}")
print(f"Desconto: {desconto*100:.0f}%")
print(f"Total final: R$ {total_final:.2f}")
```

Exercícios

1. Escreva um programa que leia um número e informe se ele é positivo, negativo ou zero.

2. Crie um programa que leia a idade de uma pessoa e classifique-a em: criança (0-12), adolescente (13-17), adulto (18-59) ou idoso (60+).
3. Faça um programa que leia dois números e um operador (+, -, *, /) e realize a operação correspondente.
4. Leia três números e imprima qual é o maior deles.

Desafios

1. Crie um programa que simule um caixa eletrônico. O usuário informa quanto quer sacar e o programa informa quantas notas de cada valor (100, 50, 20, 10, 5, 2) serão fornecidas, priorizando as notas maiores. Valide se o valor pode ser sacado com as notas disponíveis.
2. Desenvolva um programa que determine o tipo de triângulo (equilátero, isósceles, escaleno) baseado em três lados fornecidos pelo usuário. Primeiro verifique se os valores podem formar um triângulo válido.
3. Implemente um sistema de avaliação de risco de crédito que considere: renda mensal, idade, histórico de crédito (bom/ruim) e valor do empréstimo solicitado. Aprove ou negue o empréstimo baseado em critérios que você definir.

Aplicações

- **Validação de formulários**: verificar dados de entrada
- **Sistemas de autenticação**: validar credenciais
- **Jogos**: verificar condições de vitória/derrota
- **E-commerce**: aplicar regras de desconto e promoções

---

5. Laços de Repetição: `for` e `while`

Definição

Laços de repetição (ou loops) permitem executar um bloco de código várias vezes.

5.1 Laço `for`

Usado quando sabemos quantas vezes queremos repetir ou quando queremos iterar sobre uma sequência.

Sintaxe:

```
for variavel in sequencia:
    # código a ser repetido
```

Função `range()`

Gera uma sequência de números e é muito usada com `for`.

```
range(fim)           # 0 até fim-1
range(inicio, fim)   # inicio até fim-1
range(inicio, fim, passo)  # inicio até fim-1, pulando de 'passo' em 'passo'
```

Exemplos com `for`:

```
# Repetir 5 vezes
for i in range(5):
    print(f"Iteração {i}")
# Saída: 0, 1, 2, 3, 4

# Range com início e fim
for i in range(1, 6):
    print(i)
# Saída: 1, 2, 3, 4, 5

# Range com passo
for i in range(0, 11, 2):
    print(i)
# Saída: 0, 2, 4, 6, 8, 10

# Contagem regressiva
for i in range(10, 0, -1):
    print(i)
print("Feliz Ano Novo!")

# Iterar sobre string
nome = "Python"
for letra in nome:
    print(letra)

# Calcular soma
soma = 0
for i in range(1, 11):
    soma += i
print(f"Soma de 1 a 10: {soma}")
```

5.2 Laço `while`

Usado quando não sabemos quantas vezes o loop irá executar. Continua enquanto a condição for verdadeira.

Sintaxe:

```
while condicao:
    # código a ser repetido
```

Exemplos com `while`:

```
# Contador simples
contador = 0
while contador < 5:
    print(f"Contador: {contador}")
    contador += 1

# Menu de opções
opcao = ""
while opcao != "sair":
    opcao = input("Digite um comando (ou 'sair'): ")
    print(f"Você digitou: {opcao}")

# Validação de entrada
senha = ""
while senha != "1234":
    senha = input("Digite a senha: ")
    if senha != "1234":
        print("Senha incorreta. Tente novamente.")
print("Acesso concedido!")

# Loop infinito (cuidado!)
# while True:
#     print("Isso nunca para!")
```

5.3 `break` e `continue`

Comandos especiais para controlar loops.

`break`: Interrompe o loop imediatamente.

```
# Procurar um número
for i in range(1, 11):
    if i == 5:
        print("Encontrei o 5!")
        break
    print(i)
# Saída: 1, 2, 3, 4, Encontrei o 5!

# Validação com limite de tentativas
tentativas = 0
max_tentativas = 3
while tentativas < max_tentativas:
    senha = input("Digite a senha: ")
    if senha == "1234":
        print("Acesso concedido!")
        break
    tentativas += 1
    print(f"Senha incorreta. Tentativas restantes: {max_tentativas - tentativas}")
else:
    print("Número máximo de tentativas excedido.")
```

`continue`: Pula para a próxima iteração do loop.

```
# Imprimir apenas números ímpares
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(i)
# Saída: 1, 3, 5, 7, 9

# Processar apenas valores válidos
for i in range(-3, 4):
    if i == 0:
        continue  # pula divisão por zero
    print(f"10 / {i} = {10/i:.2f}")
```

Cláusula `else` em Loops

Python permite usar `else` com loops. O bloco `else` é executado quando o loop termina normalmente (sem `break`).

```
# Procurar número primo
numero = int(input("Digite um número: "))
for i in range(2, numero):
    if numero % i == 0:
        print(f"{numero} não é primo (divisível por {i})")
        break
else:
    print(f"{numero} é primo")
```

Exemplos Práticos

```
# Tabuada
numero = int(input("Tabuada do: "))
for i in range(1, 11):
    print(f"{numero} x {i} = {numero * i}")

# Fatorial
n = int(input("Calcular fatorial de: "))
fatorial = 1
for i in range(1, n + 1):
    fatorial *= i
print(f"{n}! = {fatorial}")

# Adivinhar número
import random
numero_secreto = random.randint(1, 100)
tentativas = 0
while True:
    tentativas += 1
    palpite = int(input("Adivinhe o número (1-100): "))
    if palpite == numero_secreto:
        print(f"Parabéns! Você acertou em {tentativas} tentativas!")
        break
    elif palpite < numero_secreto:
        print("Muito baixo!")
    else:
        print("Muito alto!")

# Série de Fibonacci
n = int(input("Quantos termos da sequência de Fibonacci? "))
a, b = 0, 1
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

Exercícios

1. Escreva um programa que imprima todos os números de 1 a 50.
2. Crie um programa que calcule a soma de todos os números pares entre 1 e 100.
3. Faça um programa que imprima a tabuada de um número fornecido pelo usuário (de 1 a 10).
4. Escreva um programa que leia números do usuário até que ele digite 0, então imprima a soma de todos os números digitados.
5. Crie um programa que imprima todos os números de 1 a 100, mas para múltiplos de 3 imprima "Fizz", para múltiplos de 5 imprima "Buzz", e para múltiplos de ambos imprima "FizzBuzz".

Desafios

1. Implemente um programa que verifique se um número é primo. O programa deve testar divisibilidade apenas até a raiz quadrada do número para ser mais eficiente.
2. Crie um programa que gere a sequência de Collatz: comece com um número positivo n. Se n é par, divida por 2. Se n é ímpar, multiplique por 3 e some 1. Repita até chegar em 1. Conte quantos passos foram necessários.
3. Desenvolva um jogo de pedra-papel-tesoura onde o usuário joga contra o computador. O jogo continua até que um dos jogadores vença 3 rodadas. Mantenha o placar.
4. Implemente um validador de CPF. O programa deve ler 11 dígitos e verificar se os dois dígitos verificadores estão corretos usando o algoritmo oficial de validação do CPF.

Aplicações

- **Processamento de dados**: iterar sobre registros em um banco de dados
- **Análise estatística**: calcular médias, somas, contagens
- **Jogos**: loops de jogo principal (game loop)
- **Interface de usuário**: menus interativos
- **Algoritmos**: busca, ordenação, otimização

6. Mensagens de Erro e Depuração

Definição

Erros (ou exceções) são problemas que ocorrem durante a execução do programa. **Depuração** (debugging) é o processo de identificar e corrigir esses erros.

Tipos Comuns de Erros

6.1 Erros de Sintaxe (SyntaxError)

Ocorrem quando o código não segue as regras da linguagem.

```
# Erro: falta de dois pontos
if idade > 18
    print("Maior de idade")

# Erro: parênteses não fechado
print("Olá"

# Erro: indentação incorreta
if True:
print("Erro de indentação")
```

6.2 Erros de Execução (Runtime Errors)

Ocorrem durante a execução do programa.

NameError: Variável não definida

```
print(variavel_inexistente)
# NameError: name 'variavel_inexistente' is not defined
```

TypeError: Operação inválida entre tipos

```
numero = 5
texto = "10"
soma = numero + texto
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

ValueError: Valor inapropriado

```
idade = int("vinte")
# ValueError: invalid literal for int() with base 10: 'vinte'
```

ZeroDivisionError: Divisão por zero

```
resultado = 10 / 0
# ZeroDivisionError: division by zero
```

IndexError: Índice fora do intervalo

```
lista = [1, 2, 3]
print(lista[5])
# IndexError: list index out of range
```

IndentationError: Indentação incorreta

```
def funcao():
print("Erro")
# IndentationError: expected an indented block
```

Tratamento de Erros com `try-except`

Permite capturar e tratar erros de forma controlada.

Sintaxe:

```
try:
    # código que pode gerar erro
except TipoDeErro:
    # código executado se o erro ocorrer
except OutroTipoDeErro:
    # tratar outro tipo de erro
else:
    # executado se NÃO houver erro
finally:
    # sempre executado, com ou sem erro
```

Exemplos:

```
# Tratamento básico
try:
    numero = int(input("Digite um número: "))
    resultado = 100 / numero
    print(f"Resultado: {resultado}")
except ValueError:
    print("Erro: você deve digitar um número válido")
except ZeroDivisionError:
    print("Erro: não é possível dividir por zero")

# Capturar qualquer exceção
try:
    # código arriscado
    x = 10 / 0
except Exception as e:
    print(f"Ocorreu um erro: {e}")

# Com else e finally
try:
    arquivo = open("dados.txt", "r")
    conteudo = arquivo.read()
except FileNotFoundError:
    print("Arquivo não encontrado")
else:
    print("Arquivo lido com sucesso")
    print(conteudo)
finally:
    print("Operação finalizada")
    # arquivo.close() se foi aberto

# Múltiplas exceções em uma linha
try:
    valor = int(input("Digite um número: "))
    resultado = 10 / valor
except (ValueError, ZeroDivisionError) as erro:
    print(f"Entrada inválida: {erro}")
```

Técnicas de Depuração

6.3 Uso de `print()` para Debug

A forma mais simples de depuração é imprimir valores de variáveis.

```
def calcular_media(notas):
    print(f"Debug: notas recebidas = {notas}")  # verificar entrada
    soma = sum(notas)
    print(f"Debug: soma = {soma}")  # verificar cálculo
    quantidade = len(notas)
    print(f"Debug: quantidade = {quantidade}")
    media = soma / quantidade
    print(f"Debug: media = {media}")  # verificar resultado
    return media

resultado = calcular_media([7, 8, 9])
```

6.4 Função `type()` para Verificar Tipos

```
valor = input("Digite algo: ")
print(f"Tipo da variável: {type(valor)}")  # sempre será <class 'str'>

numero = int(valor)
print(f"Tipo após conversão: {type(numero)}")  # <class 'int'>
```

6.5 Validação de Entrada

```
# Validação com loop
while True:
    try:
        idade = int(input("Digite sua idade: "))
        if idade < 0 or idade > 150:
            print("Idade inválida. Digite um valor entre 0 e 150.")
            continue
        break  # entrada válida, sair do loop
    except ValueError:
        print("Erro: digite apenas números inteiros.")

print(f"Idade registrada: {idade}")

# Função de validação reutilizável
def ler_inteiro(mensagem, minimo=None, maximo=None):
    while True:
        try:
            valor = int(input(mensagem))
            if minimo is not None and valor < minimo:
                print(f"Valor deve ser maior ou igual a {minimo}")
                continue
            if maximo is not None and valor > maximo:
                print(f"Valor deve ser menor ou igual a {maximo}")
                continue
            return valor
        except ValueError:
            print("Erro: digite um número inteiro válido")

idade = ler_inteiro("Digite sua idade: ", 0, 150)
```

6.6 Asserções (assertions)

Verificações que devem ser sempre verdadeiras. Úteis durante desenvolvimento.

```
def calcular_raiz_quadrada(numero):
    assert numero >= 0, "Número deve ser não-negativo"
    return numero ** 0.5

# Funciona
print(calcular_raiz_quadrada(16))  # 4.0

# Gera AssertionError
# print(calcular_raiz_quadrada(-4))  # AssertionError: Número deve ser não-negativo
```

Estratégias de Depuração

1. **Leia a mensagem de erro**: Python fornece informações úteis sobre o que deu errado
2. **Identifique a linha do erro**: A mensagem mostra onde o erro ocorreu
3. **Verifique os valores das variáveis**: Use `print()` antes da linha com erro
4. **Simplifique o código**: Comente partes do código para isolar o problema
5. **Teste com valores conhecidos**: Use entradas simples para verificar a lógica
6. **Execute passo a passo**: Acompanhe a execução linha por linha

Exemplos Práticos

```
# Calculadora robusta
def calculadora():
    while True:
        try:
            num1 = float(input("Primeiro número: "))
            operador = input("Operador (+, -, *, /): ")
            num2 = float(input("Segundo número: "))

            if operador == "+":
                resultado = num1 + num2
            elif operador == "-":
                resultado = num1 - num2
            elif operador == "*":
                resultado = num1 * num2
            elif operador == "/":
                if num2 == 0:
                    print("Erro: divisão por zero não é permitida")
                    continue
                resultado = num1 / num2
            else:
                print("Operador inválido")
                continue

            print(f"Resultado: {resultado}")

            continuar = input("Fazer outro cálculo? (s/n): ")
            if continuar.lower() != 's':
                break

        except ValueError:
            print("Erro: digite apenas números válidos")
        except Exception as e:
            print(f"Erro inesperado: {e}")

# calculadora()

# Processamento de lista com validação
def processar_notas():
    notas = []
```

```
    print("Digite as notas (digite 'fim' para encerrar):")

    while True:
        entrada = input("Nota: ")

        if entrada.lower() == 'fim':
            break

        try:
            nota = float(entrada)
            if nota < 0 or nota > 10:
                print("Nota deve estar entre 0 e 10")
                continue
            notas.append(nota)
        except ValueError:
            print("Entrada inválida. Digite um número ou 'fim'")

    if len(notas) == 0:
        print("Nenhuma nota foi inserida")
        return

    media = sum(notas) / len(notas)
    print(f"\nResumo:")
    print(f"Notas: {notas}")
    print(f"Quantidade: {len(notas)}")
    print(f"Média: {media:.2f}")
    print(f"Maior nota: {max(notas)}")
    print(f"Menor nota: {min(notas)}")

# processar_notas()
```

Mensagens de Erro Comuns e Soluções

Erro	Causa	Solução
`SyntaxError: invalid syntax`	Erro de sintaxe (dois pontos, parênteses, etc)	Revisar a linha indicada
`IndentationError`	Indentação incorreta	Usar 4 espaços consistentemente
`NameError: name 'x' is not defined`	Variável não foi criada	Definir a variável antes de usar
`TypeError: unsupported operand type(s)`	Operação entre tipos incompatíveis	Converter tipos ou verificar lógica
`ValueError: invalid literal`	Conversão de tipo falhou	Validar entrada antes de converter
`ZeroDivisionError`	Divisão por zero	Verificar denominador antes de dividir
`IndexError: list index out of range`	Acessar índice inexistente	Verificar tamanho da lista

Exercícios

1. Escreva um programa que leia um número do usuário e trate o caso em que ele digite texto ao invés de número.
2. Crie uma função que converta Celsius para Fahrenheit e trate possíveis erros de entrada.
3. Desenvolva um programa que leia números até o usuário digitar "sair" e calcule a média. Trate entradas inválidas.
4. Implemente um programa que leia um índice e acesse uma lista, tratando o IndexError caso o índice seja inválido.

Desafios

1. Crie um sistema de cadastro que leia nome, idade e email. Valide:
   - Nome: não pode ser vazio
   - Idade: deve ser inteiro entre 0 e 150
   - Email: deve conter "@" e "."

O programa deve continuar pedindo as informações até que todas sejam válidas.

1. Implemente uma calculadora de divisão que:
   - Trate divisão por zero
   - Trate entradas não numéricas
   - Permita múltiplas operações
   - Mantenha histórico das últimas 5 operações
   - Tenha uma opção de "desfazer" a última operação

2. Desenvolva um leitor de arquivo que tente abrir um arquivo cujo nome é fornecido pelo usuário. Trate os casos:
   - Arquivo não encontrado
   - Permissão negada
   - Arquivo vazio
   - Formate a saída de forma legível

Aplicações

- **Validação de formulários**: garantir que dados inseridos são válidos
- **Sistemas estáveis**: evitar falhas durante a execução
- **APIs**: retornar erros significativos aos usuários
- **Processamento de dados**: lidar com dados corrompidos ou ausentes
- **Aplicações críticas**: garantir funcionamento contínuo mesmo com problemas

---

Projeto Integrador

Sistema de Gerenciamento de Notas Escolares

Desenvolva um sistema completo que integre todos os conceitos aprendidos.

Requisitos:

1. Menu principal com opções:
   - Cadastrar aluno (nome e idade)
   - Registrar notas de um aluno
   - Calcular média de um aluno
   - Listar todos os alunos e suas médias
   - Verificar aprovação/reprovação (média >= 7)
   - Sair

2. Validações:
   - Nome não pode ser vazio
   - Idade deve ser inteiro positivo
   - Notas devem estar entre 0 e 10
   - Não permitir duplicação de alunos

3. Tratamento de erros:
   - Entradas inválidas
   - Tentativa de acessar aluno inexistente
   - Divisão por zero (se não houver notas)

4. Funcionalidades extras (desafio):
   - Salvar dados em arquivo de texto
   - Carregar dados ao iniciar
   - Estatísticas da turma (média geral, melhor e pior aluno)
   - Remover aluno do sistema

Estrutura sugerida:

```
# Dicionário para armazenar alunos e suas notas
# alunos = {
#     "João": {"idade": 20, "notas": [8.5, 7.0, 9.0]},
#     "Maria": {"idade": 19, "notas": [6.0, 7.5]}
# }

def menu_principal():
    # Implementar menu
    pass

def cadastrar_aluno():
    # Implementar cadastro com validação
    pass

def registrar_notas():
    # Implementar registro de notas
    pass

def calcular_media(notas):
    # Calcular e retornar média
    pass

def listar_alunos():
    # Listar todos os alunos com suas informações
    pass

def verificar_aprovacao():
    # Verificar status de aprovação
    pass

# Programa principal
# alunos = {}
# menu_principal()
```

Resumo dos Conceitos

Variáveis e Tipos

- Variáveis armazenam dados
- Python tem tipagem dinâmica
- Tipos principais: `int`, `float`, `bool`, `str`
- Use `type()` para verificar o tipo
- Use conversão explícita quando necessário

Entrada e Saída

- `input()` sempre retorna string
- `print()` exibe informações
- Use f-strings para formatação: `f"Texto {variavel}"`
- Converta tipos conforme necessário

Operadores

- Aritméticos: `+`, `-`, `*`, `/`, `//`, `%`, `**`
- Comparação: `==`, `!=`, `>`, `<`, `>=`, `<=`
- Lógicos: `and`, `or`, `not`
- Atribuição: `=`, `+=`, `-=`, etc.

Controle de Fluxo

- `if`: executa bloco se condição for verdadeira
- `elif`: testa condição alternativa
- `else`: executa se nenhuma condição anterior for verdadeira
- Indentação é obrigatória

Laços

- `for`: itera número conhecido de vezes ou sobre sequências

- `while`: repete enquanto condição for verdadeira
- `break`: interrompe o loop
- `continue`: pula para próxima iteração
- `range()`: gera sequências numéricas

Erros e Depuração

- Erros de sintaxe: corrigir antes de executar
- Erros de execução: usar `try-except`
- `print()` para debug
- Validar entradas do usuário
- Ler mensagens de erro cuidadosamente

Boas Práticas

1. **Nomenclatura**:
   - Use nomes descritivos: `idade_usuario` ao invés de `x`
   - Use snake_case: `nome_completo` ao invés de `nomeCompleto`
   - Constantes em maiúsculas: `PI = 3.14159`

2. **Comentários**:
   ```python
   # Comentário de linha única

"""
Comentário de
múltiplas linhas
"""

# Use comentários para explicar POR QUE, não O QUE
# Bom: # Adiciona 1 porque índices começam em 0
# Ruim: # Soma 1 ao contador
```

1. **Espaçamento**:
   - Use espaços ao redor de operadores: `x = 5`, não `x=5`

   - Linha em branco entre blocos lógicos
   - 4 espaços para indentação (não tabs)

2. **Validação**:
   - Sempre valide entrada do usuário
   - Use try-except para operações arriscadas
   - Forneça mensagens de erro claras

3. **Simplicidade**:
   - Evite código excessivamente complexo
   - Quebre problemas grandes em partes menores
   - Uma função deve fazer uma coisa bem feita

Recursos Adicionais

Para Praticar

- **HackerRank**: problemas de programação por nível
- **LeetCode**: desafios algorítmicos
- **CodeWars**: exercícios gamificados
- **Python Challenge**: enigmas em Python
- **Project Euler**: problemas matemáticos/computacionais

Documentação

- Documentação oficial Python: https://docs.python.org/pt-br/3/
- Python para iniciantes: https://wiki.python.org/moin/BeginnersGuide
- PEP 8 (guia de estilo): https://pep8.org/

Próximos Passos

Após dominar estes fundamentos, estude:
- Estruturas de dados (listas, tuplas, dicionários, conjuntos)
- Funções e modularização
- Manipulação de arquivos
- Programação orientada a objetos

- Bibliotecas padrão do Python
- Desenvolvimento web (Django, Flask)
- Análise de dados (Pandas, NumPy)
- Automação de tarefas

Exercícios Finais de Revisão

1. Crie um programa que calcule o fatorial de um número usando loop `for`.
2. Desenvolva um conversor de moedas que permita conversões entre Real, Dólar e Euro, com menu de opções e validação de entrada.
3. Implemente um jogo de adivinhação onde o computador pensa em um número de 1 a 100 e o usuário tem 7 tentativas para acertar. Forneça dicas "maior" ou "menor".
4. Crie um validador de senha que verifique se uma senha tem:
   - Pelo menos 8 caracteres
   - Pelo menos uma letra maiúscula
   - Pelo menos uma letra minúscula
   - Pelo menos um número
5. Desenvolva uma calculadora de IMC que classifique o resultado em: abaixo do peso, peso normal, sobrepeso, obesidade grau I, II ou III.
6. Implemente um programa que leia uma lista de números (até digitar -1) e informe:
   - Quantidade de números
   - Soma total
   - Média
   - Maior e menor número
   - Quantos são pares e ímpares
7. Crie um conversor de temperatura que converta entre Celsius, Fahrenheit e Kelvin, com menu interativo.
8. Desenvolva um programa que verifique se uma palavra é palíndromo (lê-se igual de trás para frente).
9. Implemente um sistema de login com 3 tentativas. Após 3 falhas, bloquear o acesso.

10. Crie um programa que desenhe padrões com asteriscos:

    ```
    * ** *** **** *****
    ```

    Permita que o usuário escolha o número de linhas.
