Python Através de Exemplos

Sumário

1 Conceitos, fundamentos, saída de dados e entrada de dados

1.1 Comentários

```
# Esta linha é um comentário – uma informação que não será processada
# Esta é outra linha de comentário – também não participa da execução do programa
# Um comentário aumenta a clareza na leitura do código do programa
print(3.14159)   # Comentário na mesma linha que um comando
raiz = sqrt(16)  # Extrai a raiz quadrada de 16 e armazena na variável raiz
"""
Este é um bloco de comentário, que pode ter várias linhas
Começa com uma linha com três aspas duplas
Termina com uma linha com três aspas duplas
"""
```

1.2 Tipos de dados

O **tipo** define a forma dos dados e as operações que podem ser feitas com ele (soma, multiplicação, concatenação, ...). Python possui os tipos **int** (número inteiro), **float** (número real de ponto flutuante), **str** (cadeia de caractere ou string) e **bool** (valor lógico True ou False).

1.3 Valores constantes

```
123        # É um valor inteiro
-4760      # É um valor inteiro
0          # É um valor inteiro
3.14159    # É um valor real de ponto flutuante
```

```
0.314159e1 # É um valor real de ponto flutuante
0.0        # É um valor real de ponto flutuante
"Texto"    # É uma cadeia de caracteres
'Texto'    # É uma cadeia de caracteres
"A"        # É uma cadeia de caracteres
True       # É um valor lógico
False      # É um valor lógico
```

1.4 Identificadores

Um **identificador** é um **nome** criado para referenciar uma **variável** e outras entidades de um programa Python. Um **identificador** é uma cadeia de caracteres latinos (**A-Z**, **a-z**, **0-9**, **_**), deve começar com uma letra no intervalo "**A-Z**", "**a-z**", ou com "**_**" (caractere sublinha), nunca com um dígito.

1.5 Variáveis

Uma **variável** é um pedaço de memória onde se pode armazenar um valor inteiro, real de ponto flutuante, uma cadeia de caractere (string) ou um valor lógico (True ou False).

1.6 Operadores aritméticos

Operador	Operação
- (unário)	Inverte o sinal do valor ou da variável
+	Adição
-	Subtração
*****	Multiplicação
/	Divisão
//	Divisão inteira
%	Módulo – resto da divisão inteira

1.7 Precedência dos operadores

Da maior precedência para a menor (de cima para baixo):	
- (operador unário)	
(...(...)...)	dos mais internos para os mais externos
******	
***, /, // e %**	o que primeiro aparecer da esquerda para a direita
+ e -	o que primeiro aparecer da esquerda para a direita

1.8 Funções

1.8.1 Algumas funções prontas

Função	Descrição da função

Arredondamento	
round(x)	Soma 0.5 ao número real x e retorna a parte inteira da soma.
floor(x)	Retorna o piso de x, o maior inteiro menor ou igual a x.
ceil(x)	Retorna o teto de x, o menor inteiro maior ou igual a x.
Raízes e logaritmos	
sqrt(x)	Retorna a raiz quadrada de x.
log(x)	Com um argumento, retorna o logaritmo natural de x (para a base e). Com dois argumentos, retorna o logaritmo de x para a base fornecida.
e	A constante matemática e = 2.71828 ...
Trigonometria	
sin(x)	Retorna o seno de x radianos.
asin(x)	Retorna o arco seno de x, em radianos.
pi	A constante matemática π = 3.14159 ...
Outras	
abs(x)	Retorna o inteiro x sempre com o sinal positivo.
int(x)	Se x for real, converte x para um inteiro, perdendo a fração; sendo x uma cadeia de caracteres (string) representando um inteiro, converte x para um inteiro; sendo x um valor lógico True, retorna 1; e, sendo x um valor lógico False, retorna 0.
float(x)	Se x for inteiro, converte x para um real, com a fração zero; sendo x uma cadeia de caracteres (string) representando um real, converte x para um real; sendo x um valor lógico True, retorna 1.0; e, sendo x um valor lógico False, retorna 0.0.
str(x)	Converte o valor de x, retornando uma cadeia de caracteres (string), independentemente do tipo de x.
bool(x)	Converte x para um valor lógico: se x for um número igual a 0, retorna False; se x for um número diferente de 0, retorna True; e se x for uma cadeia de caracteres, retorna True.
Entrada e saída	
print(..., ...)	Mostra na tela uma lista de: valores, variáveis e/ou expressões.
input("texto")	Retorna um valor digitado no teclado do tipo cadeia de caracteres (string) – o "texto" é opcional e é mostrado na tela antes da digitação.

1.8.2 Importação de módulos e funções

```
import módulo              # a chamada das funções será módulo.função()
from módulo import função  # chamada da função será função()
```

1.8.3 Exemplos

```
# Importa o módulo
import math
print(math.sqrt(16))  # referencia o módulo e a função
# Importa só uma função do módulo
from math import ceil
print(ceil(16.345))   # referencia só a função
```

1.9 Expressões aritméticas

```
5+10                  # Adição de dois valores inteiros
17-2*8                # Primeiro a multiplicação, depois a subtração
(17-2)*8              # Primeiro a subtração, depois a multiplicação
1000+base**16         # O valor armazenado na variável de nome base será elevado ao
                      # expoente 16 e o resultado será somado a 1000
37/3                  # A divisão resulta no real 12.333333333333334
37//3                 # A divisão resulta no inteiro 12
37%3                  # Resulta em 1 - o resto da divisão inteira (módulo)
b**2-4*a*c            # A expressão será avaliada segundo as regras da álgebra,
                      # utilizando os valores armazenados nas variáveis de nomes
                      # a, b e c
-b+sqrt(delta)/(2*a)  # Primeiro, o sinal da varável b será invertido; em seguida,
                      # será extraída a raiz quadrada do valor armazenado na
                      # variável delta; e por fim, as demais operações serão
                      # normalmente avaliadas, utilizando o valor constante 2 e
                      # os valores armazenados nas variáveis
Obs 1: As regras da álgebra sempre prevalecerão.
Obs 2: As expressões entre parênteses sempre serão avaliadas em primeiro lugar –
       na ordem dos mais internos para os mais externos.
```

1.10 Atribuição

```
idade = 10              # Armazena na variável idade o valor inteiro 10
dinheiro = 123.45       # Armazena em dinheiro o valor real 123.45
delta = b**2-4*a*c      # Calcula o valor da expressão `a direita do igual
                        # e armazena o resultado na variável à esquerda do igual
x2 = (-b-sqrt(b**2-4*a*c))/(2*a)    # Calcula o valor da expressão e armazena o
                                    # resultado na variável x2
nome = input("Informe o seu nome")    # Mostra a mensagem (texto) na tela, recebe
                                      # uma cadeia de caracteres digitada no teclado
                                      # e armazena a cadeia na variável nome
nota1 = float(input())    # Recebe uma cadeia de caracteres digitada no teclado e
                          # armazena a cadeia na variável nome
inteiro = int(dividendo/divisor)    # Extrai a parte inteira do quociente da
                                    # divisão e armazena como inteiro na variável
                                    # inteiro
data = 11122020    # Armazena na variável de nome data, o valor inteiro com a
                   # data 11/12/2020 no formato DDMMAAAA
mes = data/10000%10    # Extrai o valor inteiro do mês, no formato MM, da variável
                       # inteira de nome data e armazena na variável mes
```

1.11 Concatenação de cadeias de caracteres

```
s = "ABC"               # Armazena a cadeia "ABC" na variável s
```

```
s+"C"                    # Resulta na cadeia "ABCC"
s+"D"*4                  # Resulta na cadeia "ABCDDDD"
"X"+"-"*10+"X"           # Resulta na cadeia "X----------X"
S+"x4 = "+s*4            # resulta na cadeia "ABCx4 = ABCABCABCABC"
nome = "Alan"            # Armazena a cadeia "Alan" na variável nome
sobrenome = "Turing"     # Armazena a cadeia "Turing" na variável sobrenome
nome+" "+sobrenome       # Resulta na cadeia "Alan Turing"
```

1.12 Composição de cadeia de caracteres (strings)

A composição de uma cadeia de caracteres (string) é a intercalação de uma cadeia com outras cadeias e/ou outros tipos de dados (inteiro, real de ponto flutuante ou lógico.

1.12.1 Marcadores de posições

Marcador	Tipo
%d	Número inteiro
%f	Número real de ponto flutuante
%s	Cadeia de caracteres (string) ou valor lógico

1.12.2 Método de composição com marcadores de posições

```
from math import pi                 # importa a constante pi do módulo matemático
inteiro = 123                       # Armazena 123 na variável inteiro
real = 123.456789                   # Armazena 123.456789 na variável real
cad_real= "real"                    # Armazena a cadeia "real" na variável cad_real
"Valor de pi = %f" % pi             # Resulta na cadeia "Valor de pi = 3.141593"
"Valor de inteiro = %d" % inteiro   # Resulta na cadeia "Valor de inteiro = 123"
"Valor de real = %f" % real         # Resulta na cadeia
                                    # "Valor de real = 123.456789"
# As três composições abaixo resultam na cadeia "inteiro = 123 e real = 123.456"
"inteiro = %d e real = %.3f" % (inteiro, real)
"%s = %d e %s = %.3f" % ("inteiro", inteiro, "real", real)
"%s = %d e %s = %.3f" % ("inteiro", inteiro, cad_real, real)
"Verdadeiro = %s" % True    # Resulta na cadeia "Verdadeiro = True"
"Valor lógico = %s ou %s" % (True, False)    # Resulta na cadeia
                                             # "Valor lógico = True ou False"
"[%d]" % inteiro      # Resulta na cadeia "[123]"
"[%5d]" % inteiro     # Resulta na cadeia "[  123]"
"[%-5d]" % inteiro    # Resulta na cadeia "[123  ]"
"[%05d]" % inteiro    # Resulta na cadeia "[00123]"
"[%f]" % 5            # Resulta na cadeia "[5.000000]"
"[%f]" % real         # Resulta na cadeia "[123.456789]"
"[%.3f]" % real       # Resulta na cadeia "[123.456]"
"[%7.3f]" % real      # Resulta na cadeia "[123.456]"
"[%9.2f]" % real      # Resulta na cadeia "[   123.45]"
```

```
"[%-9.2f]" % real     # Resulta na cadeia "[123.45   ]"
"[%09.2f]" % real     # Resulta na cadeia "[000123.45]"
```

1.12.3 Método de composição com format

```
from math import pi                 # importa a constante pi do módulo matemático
inteiro = 123                       # Armazena 123 na variável inteiro
real = 123.456789                   # Armazena 123.456789 na variável real
cad_real= "real"                    # Armazena a cadeia "real" na variável cad_real
"Valor de pi = {}".format(pi)       # Resulta na cadeia
                                    # "Valor de pi = 3.141592653589793"
"Valor inteiro = {}".format(inteiro)     # Resulta na cadeia
                                         # "Valor de inteiro = 123"
"Valor real = {}".format(real)    # Resulta na cadeia "Valor de real = 123.456789"
# As três composições abaixo resultam na cadeia "inteiro = 123 e real = 123.456"
"inteiro = {:d} e real = {:.3f}".format(inteiro, real)
"{} = {:3d} e {} = {:7.3f}".format("inteiro", inteiro, "real", real)
"{:s} = {:d} e {} = {:.3f}".format("inteiro", inteiro, cad_real, real)
"Verdadeiro = {}".format(True)    # Resulta na cadeia "Verdadeiro = True"
"Valor lógico = {} ou {}".format(True, False)    # Resulta na cadeia
                                                 # "Valor lógico = True ou False"
"[{}]".format(inteiro)         # Resulta na cadeia "[123]"
"[{:d}]".format(inteiro)       # Resulta na cadeia "[123]"
"[{:5d}]".format(inteiro)      # Resulta na cadeia "[  123]"
"[{:<5}]".format(inteiro)      # Resulta na cadeia "[123  ]"
"[{:05}]".format(inteiro)      # Resulta na cadeia "[00123]"
"[{}]".format(5)               # Resulta na cadeia "[5.000000]"
"[{:f}]".format(real)          # Resulta na cadeia "[123.456789]"
"[{:.3}]".format(real)         # Resulta na cadeia "[123.456]"
"[{:7.3f}]".format(real)       # Resulta na cadeia "[123.456]"
"[{:9.2f}]".format(real)       # Resulta na cadeia "[   123.45]"
"[{:<9.2}]".format(real)       # Resulta na cadeia "[123.45   ]"
"[{:^10.2f}]".format(real)     # Resulta na cadeia "[  123.45  ]"
"[{:09.2}]".format(real)       # Resulta na cadeia "[001.2e+02]"
"[{:>10s}]".format("texto")    # Resulta na cadeia "[     texto]"
```

1.12.4 Método de composição com f-string

```
from math import pi      # importa a constante pi do módulo matemático
inteiro = 123            # Armazena 123 na variável inteiro
real = 123.456789        # Armazena 123.456789 na variável real
cad_real= "real"         # Armazena a cadeia "real" na variável cad_real
logico = True            # Armazena o valor lógico True na variável logico
```

```
f"Valor de pi = {pi}"    # Resulta na cadeia "Valor de pi = 3.141592653589793"
f"Valor inteiro = {inteiro}"    # Resulta na cadeia "Valor de inteiro = 123"
f"Valor real = {real}"          # Resulta na cadeia "Valor de real = 123.456789"
# As três composições abaixo resultam na cadeia "inteiro = 123 e real = 123.456"
f"inteiro = {inteiro} e real = {real:.3f}"
f"inteiro = {inteiro:3d} e real = {real:7.3f}"
f"inteiro = {inteiro:d} e {cad_real} = {real:.3f}"
f"Verdadeiro = {True}"    # Resulta na cadeia "Verdadeiro = True"
f"Valor lógico = {logico} ou {False}"    # Resulta na cadeia
                                         # "Valor lógico = True ou False"
f"[{inteiro}]"          # Resulta na cadeia "[123]"
f"[{inteiro:d}]"        # Resulta na cadeia "[123]"
f"[{inteiro:5d}]"       # Resulta na cadeia "[  123]"
f"[{inteiro:<5}]"       # Resulta na cadeia "[123  ]"
f"[{inteiro:05}]"       # Resulta na cadeia "[00123]"
f"[{5}]"                # Resulta na cadeia "[5.000000]"
f"[{real:f}]"           # Resulta na cadeia "[123.456789]"
f"[{real:.3}]"          # Resulta na cadeia "[123.456]"
f"[{real:7.3f}]"        # Resulta na cadeia "[123.456]"
f"[{real:9.2f}]"        # Resulta na cadeia "[   123.45]"
f"[{real:<9.2}]"        # Resulta na cadeia "[123.45   ]"
f"[{real:^10.2f}]"      # Resulta na cadeia "[  123.45  ]"
f"[{real:09.2}]"        # Resulta na cadeia "[001.2e+02]"
f"[{cad_real:>10s}]"    # Resulta na cadeia "[      real]"
```

1.13 Saída de dados

```
print(..., ...)
```

1.13.1 Sem formatação

```
print(5)                        # Imprime: 5
print(3.14159)                  # Imprime: 3.14159
print(31415.9e-4)               # Imprime: 3.14159
print("texto")                  # Imprime: texto
print(5, -3.14159, "texto")     # Imprime: 5 -3.14159 texto
print(5+10)                     # Imprime: 15
print(3*7, (17-2)*8)            # Imprime: 21 120
print(2**16)                    # Imprime: 65536
dividendo = 37                  # Armazena o inteiro 37 na variável dividendo
print(dividendo/3)              # Imprime: 12.333333333333334
print(dividendo//3)             # Imprime: 12
print(dividendo%3)              # Imprime: 1
```

1.13.2 Com formatação

```
print("composição de cadeia de caracteres")
```

1.13.3 Na mesma linha

```
print("T", end = "")  # Imprime T sem mudar de linha
print("e", end = "")  # Imprime e sem mudar de linha
print("x", end = "")  # Imprime x sem mudar de linha
print("t", end = "")  # Imprime t sem mudar de linha
print("o")            # Imprime o e muda de linha
```

1.13.4 Em linhas diferentes com um mesmo comando

```
# No print() abaixo: muda de linha toda vez que encontra um '\n' (caractere 'nova
# linha' ou 'new line'), mesmo com um único elemento a ser impresso
#
print("T\ne\n\nx\n\n\nt\n\n\n\no")
#
# No print() abaixo: insere um espaço (' ') entre os elementos impressos e cada
# '\n' (caractere 'nova linha' ou 'new line') encontrado provoca uma mudança de
# linha
#
print("T", "\n", "e", "\n\n", "x", "\n\n\n", "t", "\n\n\n\n", "o")
```

1.14 Entrada de dados

```
input()
```

1.14.1 Sem prompt

```
texto = input()           # Recebe do teclado uma cadeia de caracteres e armazena
                          # na variável de nome texto
inteiro = int(input())    # Recebe do teclado uma cadeia de caracteres no formato
                          # de um inteiro, converte-a para inteiro e armazena na
                          # variável de nome inteiro
real = float(input())     # Recebe do teclado uma cadeia de caracteres no formato
                          # de um real, converte-a para ponto flutuante e armazena
                          # na variável de nome real
```

1.14.2 Com prompt

```
nome = input("Informe o seu nome: ")    # Mostra a mensagem, recebe do teclado uma
                                        # cadeia de caracteres e armazena na
                                        # variável de nome nome
idade = int(input("Qual é a sua idade? "))    # Mostra a mensagem, recebe do
                                              # teclado uma cadeia de caracteres no
                                              # formato de um inteiro, converte-a
                                              # para inteiro e armazena na variável
                                              # de nome idade
peso = float(input("Informe o seu peso: "))    # Mostra a mensagem, recebe do
                                               # teclado uma cadeia de caracteres
                                               # no formato de um real, converte-a
                                               # para ponto flutuante e armazena na
                                               # variável de nome peso
```

2 Comandos condicionais

2.1 O tipo bool

O tipo em Python para armazenar os valores lógicos é chamado de **bool**. Existem apenas dois valores booleanos: True (verdadeiro) e False (falso) – as iniciais maiúsculas são obrigatórias.

2.2 Operadores relacionais e lógicos

Operador	Operação
Operadores Relacionais	
<	Menor que
<=	Menor ou igual a
>	Maior que
<=	Menor ou igual a
==	Igual a
!=	Não igual a (diferente de)
Operadores Lógicos	
and	Verdadeiro se ambas as condições forem verdadeiras
or	Verdadeiro se pelo menos uma condição for verdadeira
not	Verdadeiro se a expressão for falsa e falsa caso contrário

2.3 Expressões relacionais e lógicas

```
x != y                # verdadeiro quando x é diferente de y
x > y                 # verdadeiro quando x é maior do que y
x < y                 # verdadeiro quando x é menor do que y
x >= y                # verdadeiro quando x é maior ou igual a y
x <= y                # verdadeiro quando x é menor ou igual a y
x == 5                # verdadeiro quando x é igual a 5
x>0 and x<10          # verdadeiro quando x está entre 1 e 9
n%2 == 0 or m%2 == 0  # verdadeiro quando n ou m for par
not(b%a)              # verdadeiro quando a é divisor de b
```

2.4 Precedência de operadores

Nível	**Categoria**	**Operadores**
9(alto)	Sinal	- (unário)
8	Parênteses	(...(...)...)
7	exponenciação	**
6	Multiplicação e divisão	*, /, //, %
5	Adição e subtração	+, -
4	relacional	==, !=, <=, >=, >, <
3	lógico	**not**
2	lógico	**and**
1(baixo)	lógico	**or**

2.5 Comando condicional if

2.5.1 Sintaxe

```
if EXPRESSÃO BOOLEANA:
    COMANDOS_1        # executados se condição tem valor True
```

2.5.2 Exemplo

```
x = 10
if x < 0:
    print("O numero negativo ",  x, " não é valido aqui.")
print("Isto e' sempre impresso.")
```

2.6 Comando condicional if-else

2.6.1 Sintaxe

```
if EXPRESSÃO BOOLEANA:
    COMANDOS_1        # executados se condição tem valor True
else:
    COMANDOS_2        # executados se condição tem valor False
```

2.6.2 Exemplo

```
f x < 0:
    print("O número", x, "é negativo.")
else:
    print(x, " é um número positivo ou zero.")
print("Isto é sempre impresso.")
```

2.7 Comando condicionais aninhados

```
São comandos condicionais combinados – uns dentro de outros
```

2.8 Exemplo

```
if x < y:
    print("x e' menor do que y.")
else:
    if x > y:
        print("x e' maior do que y.")
    else:
        print("x e y devem ser iguais.")
```

2.9 Comando condicional if-elif-else

2.9.1 Sintaxe

```
if EXPRESSÃO_BOOLEANA_1:
    COMANDOS_1        # executados se condição tem valor True
elif EXPRESSÃO_BOOLEANA_2:
    COMANDOS_2        # executados se condição tem valor True
...
elif EXPRESSÃO_BOOLEANA_n:
    COMANDOS_n        # executados se condição tem valor True
```

```
else:
    COMANDOS_f        # executados se todas as condições têm valor False
```

2.9.2 Exemplo

```
if x < y:
    print("x e' menor do que y.")
elif x > y:
    print("x e' maior do que y.")
else:
    print("x e y devem ser iguais.")
```

3 Comandos de repetição