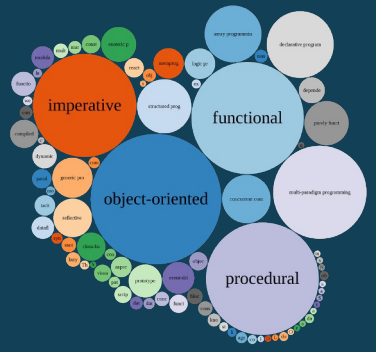
Paradigmas de Linguagem de Programação em Python



Tipos de Dados, Variáveis, Expressões e I/O

introducao.ipynb

print()
Saída/Exibição de Dados

A função **`print()`**, que, em inglês, significa **imprimir**, imprime na tela o que estiver entre parênteses.

Utilizaremos sempre que quisermos mostrar algo, como uma mensagem ou o resultado de uma operação numérica, etc.

Assim como na matemática, em que **`f(x)`** opera em **`x`**, que está entre parênteses.

Variáveis

Qualquer dado que seja manipulado pelo programa deve ficar armazenado na memória principal do computador.

Para que o armazenamento de dados seja possível, é preciso reservar espaços para isso na memória principal.

As variáveis e constantes são unidades básicas de armazenamento dos dados em programação.

Elas são um espaço de memória reservado para armazenar um certo tipo de dado e possuem um identificador (nome) para referenciar o seu conteúdo.

Variáveis

Podemos **nomear** regiões da memória RAM para facilitar nosso trabalho.

Podemos fazer **a = 5** para armazenar, em algum local da RAM, o valor **5**. Podemos dizer que **"a recebe 5"**.

Podemos nos referir a esse local utilizando a **variável** de nome **a**.

A importância do uso das variáveis reside, principalmente, na inteligibilidade e no reuso do código.

Ex.:
```
a = 5
b = 3
print(a + b)
```


5

Variáveis
Declarações Múltiplas

No Python, podemos declarar diversas variáveis de uma só vez.

Podemos, também:

- Atribuir o mesmo valor a mais de uma variável diferente
- Misturar os tipos de variáveis

6

Variáveis
Declarações Múltiplas

```
aula1, aula2, aula3 = "Matemática", "Química", "Física"

print(aula1)
print(aula2)
print(aula3)
```


7

Variáveis
Declarações Múltiplas

```
assunto1 = assunto2 = assunto3 = "Teoria da relatividade"

print(assunto1)
print(assunto2)
print(assunto3)
```


8

Variáveis
Declarações Múltiplas

```
idade, nome, vivo = 30, ”Henrique", True

print("idade:", idade)
print("nome:", nome)
print("vivo:", vivo)
```


9

Nome de variáveis

Deve começar com **letra** ou **sublinhado** (_).

Somente deve conter caracteres **alfanuméricos** ou **sublinhado**.

Os nomes são sensíveis a maiúsculas e minúsculas.

Evite copiar **palavras reservadas**.

Nome de Varíaveis
Palavras Reservadas (regras)

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```

Nome de variáveis

```
a1 = 3                  # ok!
_nome = "elpidio"       # ok!
print = 4.1             # evitar!
1a = 5                  # errado!
f.t = 17                # errado!
p@m = 0                 # errado!
```

Variáveis: Tipos de Dados

As variáveis têm, além de valores e nomes, **tipos**.

Python possui **tipagem dinâmica**: não precisamos especificar os tipos das variáveis, como em outras linguagens.

Os tipos são:

- Texto: `str`
- Numéricos: `int`, `float`, `complex`
- Sequência: `list`, `tuple`, `range`
- Mapeamento: `dict`
- Conjuntos: `set`, `frozenset`
- Booleano: `bool`
- Binários: `bytes`, `bytearray`, `memoryview`

Tipos de Dados
numerais (reais, naturais)

Armazena valores **inteiros** ou de **ponto flutuante**

Números com ponto flutuante ou decimais são aqueles com parte decimal (1.0, 5.1, 10.6)

Note que utilizamos o ponto (.) e não a vírgula

Operadores Aritméticos
+, -, *, /, **, %, //

Operador	Operação	Exemplo	Resultado
+	Adição	2 + 7	9
-	Subtração	14.5 - 0.5	14
*	Multiplicação	2 * 18	36
/	Divisão	10 / 5	2
**	Exponenciação	2 ** 10	1024
%	Resto de divisão inteira	13 % 2	1
//	Quociente da Divisão	10 // 3	3

Operadores Aritméticos
Operação com Atribuição
1/2

Operador	Exemplo	Operação
=	x = 5	x recebe 5
+=	x += 5	Adiciona 5 a x
-=	x -= 3	Decrementa 3 de x
*=	x *= 2	Multiplica x por 2
/=	x /= 4	Divide x por 4

Operadores Aritméticos
Operação com Atribuição
2/2

Operador	Exemplo	Operação
**=	x **= 2	x elevado a 2
%=	x %= 2	Resto de x por 2
//=	x //= 3	Quociente inteiro de x por 3

Tipos de Dados
boolean (lógicos)

São **binárias**:

True (1) ou

False (0)

Podem surgir a partir de operadores relacionais e lógicos

Operadores relacionais
==, >, <, !=, >=, <=

Operador	Operação
==	Igualdade
>	Maior que
<	Menor que
!=	Diferente
>=	Maior ou igual
<=	Menor ou igual

Operadores lógicos
and, or, not

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

Operadores lógicos (bit a bit)
and (&), or (|), xor (^) ...

< operando 1 > **&** < operando 2>

```
a = 30
b = 7
a & b = ?
```

```
30  →  1 1 1 1 0
 7  →  0 0 1 1 1
30 & 7  →  0 0 1 1 0 = 6
```

Tabela Verdade		
A	**B**	**A & B**
1	1	1
1	0	0
0	1	0
0	0	0

Operadores lógicos (bit a bit)
and (&), **or (|)**, xor (^) ...

< operando 1 > | < operando 2>

```
a = 30
b = 7
a | b = ?
```

```
30  →  1 1 1 1 0
 7  →  0 0 1 1 1
30 | 7  →  1 1 1 1 1 = 31
```

Tabela Verdade		
A	**B**	**A | B**
1	1	1
1	0	1
0	1	1
0	0	0

Operadores lógicos (bit a bit)

and (&), or (|), **xor (^)** ...

```
< operando 1 > ^ < operando 2>
a = 30
b = 7
a ^ b = ?
```

```
30  →  1 1 1 1 0
 7  →  0 0 1 1 1
30 ^ 7  →  1 1 0 0 1 = 25
```

Tabela Verdade		
A	**B**	**A ^ B**
1	1	0
1	0	1
0	1	1
0	0	0

Operadores lógicos (bit a bit)
and (&), or (|), xor (^) ...

A	B	A and B	A or B	A xor B
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

Operadores Lógicos
Operação com Atribuição

Operador	Exemplo	Igual a
&=	x &= 2	x = x & 2
|=	x |= 2	x = x | 2
^=	x ^= 3	x = x ^ 2

Tipos de Dados
strings (textuais)

São cadeias de caracteres

São declaradas com aspas duplas

Possuem algumas **funções** para ajudar na manipulação:

- len() → retorna o tamanho
- ...

Seus elementos podem ser acessados com uso de []

- Operador de indexação

Tipos de Dados
strings (textuais)

Operações que podem ser realizadas:

Particionamento nome[1:4]

Concatenação: "aa" + "bb"

Formatação: "fiz %d marinheiros" % 10

- %d → numeros inteiros
- %f → numeros reais
- %s → strings

input()
Entrada de dados

Forma de capturar informações do usuário

A função recebe uma **string**, que é uma mensagem, e retorna um valor, capturado após o usuário digitar *enter*

Sempre retorna uma string

- Mesmo se a entrada for um número, será processada como string
- Para forçar, devemos fazer um **cast (alterar o tipo para int, float, etc)**

input()
Entrada de dados

Sempre retorna uma string

Ponto frágil → necessário validar os dados capturados

Nada impede que o usuário forneça caracteres aleatórios ao programa: números em vez de textos, ou vice-versa

Resultado comum → Erros

Conversão de tipos (cast) Números

Podemos fazer conversões de tipos de números utilizando as seguintes funções:

```
float()
int()
hex()
bin()
```

Conversão de tipos (cast)
Números

```
print(float(3))
print(int(1.0))
print(int(1.682))

print(hex(77))
print(bin(77))
```

Conversão de tipos (cast)
Texto

Podemos fazer conversões para **`string`** utilizando a seguinte função:

`str()`

```
valor = 3
x = str(valor)

print(x)
```

Obrigado!

Alguma dúvida?