



Laboratório de Programação Competitiva

Profa. Silvia Brandão

2024.2

Aula de hoje ...

- Revisando as características do Python.
- Explorando os conceitos de: variáveis, tipos de dados, entrada e saída de dados, Expressões e operadores (aritméticos; relacionais e lógicos), strings e string formatada.
- Estruturas de dados em Python: listas, tuplas, dicionários e conjuntos.
- Discussão sobre estratégias de resolução de código.



Não se esqueçam do Uniube+

Dica de leitura:

<https://wiki.python.org.br/PythonInstantaneo>

LINGUAGEM PYTHON, por que?











Figura 2 – Guido Van Rossum, criador da linguagem Python.
Fonte: https://pt.wikipedia.org/wiki/Guido_van_Rossum.



- Linguagem de código aberto (Open Source), escrita por Guido Van Rossum, em 1989;
- Linguagem de programação de alto nível;
- Conhecida por ser interpretada, multiplataforma, multiparadigma e extremamente versátil;
- Possui sintaxe simples e legibilidade excepcional, apresentando uma curva de aprendizagem pequena (facilidade para aprender, ler e compreender);
- Grandes empresas utilizando e investindo em melhorias, acarretando ampla variedade de uso;
- Interpretada e Orientada a Objetos desde sua criação;
- Uso de indentação para marcação de blocos;
- Quase nenhum uso de palavras-chave associadas com compilação;
- Possuir coletor de lixo para gerenciar automaticamente o uso de memória;
- Compete com linguagens de uso geral como Java e C++, apresentando uma produtividade elevada;
- Se destaca no desenvolvimento de aplicações em ciência de dados e inteligência artificial;
- Oferece estrutura de dados de alto nível: strings , listas, tuplas, dicionários, arquivos, classes;
- Possui uma extensa biblioteca de módulos e pacotes, proporcionando eficiência e produtividade no desenvolvimento de projetos.

Índice TIOBE Programming Community - é um indicador da popularidade das linguagens de programação



Aug 2024	Aug 2023	Change	Programming Language		Ratings	Change
1	1			Python	18.04%	+4.71%
2	3	▲		C++	10.04%	-0.59%
3	2	▼		C	9.17%	-2.24%
4	4			Java	9.16%	-1.16%
5	5			C#	6.39%	-0.65%
6	6			JavaScript	3.91%	+0.62%
7	8	▲		SQL	2.21%	+0.68%
	7	▼		Visual Basic	2.18%	-0.45%



LINGUAGEM PYTHON – interpretada?

SIM!

Python é uma linguagem de programação interpretada.

A compilação é um processo intermediário.

O código escrito em Python é compilado para um **Bytecode**, que é uma representação mais baixo nível, menos compreensível por humanos, que será lido pelo interpretador.

Ela é considerada interpretada porque o código gerado após a compilação não é código de máquina, mas sim um **código intermediário**, então não pode ser executado diretamente pela máquina, mas sim pelo interpretador.

Variáveis

Exemplos de variáveis:

```
meu_nome  
numero_de_cadastro  
telefone_residencial  
soma1  
a23
```

- O nome de uma variável começa sempre com uma letra do alfabeto;
- Números são permitidos no nome da variável, desde que a partir do segundo caractere;
- Espaço entre dois nomes definem duas variáveis distintas;
- Quando declaradas, as variáveis são associadas a um tipo e a um escopo;
- O escopo da variável é definido pela indentação.

```
var = 9  
print(type(var))  
var = 'tudo bem?'  
print(type(var))
```

após execução →

```
<class 'int'>  
<class 'str'>
```

Entrada e Saída de dados:

```
# Declaração de variáveis  
mensagem = "O que há velhinho?"  
n = 17  
pi = 3.14159  
nome = input("Digite seu nome ")  
  
print(mensagem)  
print(n)  
print(pi)  
print(nome)
```

TIPAGEM DINÂMICA

Uma linguagem é classificada como possuidora da **tipagem dinâmica**, quando o tipo de dados de sua variável é a ela atribuído somente por ser **inicializada** com um valor qualquer.

Reflita sobre os resultados impressos.

Snake case e Camel-case

- O Python possui um **guia de estilos e estruturação** que fornece convenções de codificação para o código, o PEP8. Isso não significa que seu código não funcionará se não seguir as diretrizes, no entanto, segundo o PEP20, *The Zen of Python*, a legibilidade conta.
- **snake_case** para variáveis, funções e métodos;
- **PascalCase** para classes;
- **SCREAMING_SNAKE_CASE** para constantes.

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

    def exibir_primeiro_nome(self):
        print(self.nome)

pessoa_um = Pessoa('Aida', '123456789')
```

Operadores em Python

OPERADORES ARITMÉTICOS

OPERADOR	UTILIZAÇÃO	EXEMPLO
+	Adição	$2 + 2 = 4$
-	Subtração	$4 - 2 = 2$
*	Multiplicação	$6 * 3 = 18$
/	Divisão	$8 / 2 = 4$
**	Potência	$8 ** 2 = 64$
%	Módulo	$4 \% 2 = 0$

OPERADORES DE COMPARAÇÃO

OPERADOR	UTILIZAÇÃO	EXEMPLO
==	Igualdade	$a == 38$
!=	Diferença	$b != 3$
<	Menor que	$c < 8$
>	Maior que	$d > 6$
<=	Menor ou Igual	$b <= 16$
>=	Maior ou Igual	$k <= b$
in	Verifica a presença de um elemento em uma sequência	$b \text{ in lista}$

Operadores em Python

OPERADORES DE ATRIBUIÇÃO

OPERADOR	UTILIZAÇÃO	EXEMPLO
=	Atribuição	x = 2
+=	Soma	x += 2 (x=x+2)
-=	Subtração	x -= 2 (x=x-2)
*=	Multiplicação	x *= 2 (x=x*2)
/=	Divisão	x /= 2 (x=x/2)
//=	Divisão inteira	x //= 2 (x=x//2)
=	Potência	x **= 2 (z = z2)

OPERADORES LÓGICOS

OPERADOR	UTILIZAÇÃO	EXEMPLO
AND	Conjunção	(a and b)
OR	Disjunção	(k or j)
XOR	Disjunção exclusiva	(w xor p)
NOT	Negação	not (a+b)

OPERADORES SOBRE STRINGS

+	Concatenação	'lua' + 'azul'
*	Repetição	'a' * 5

Tipos de dados

TIPOS DE DADOS	
Nome	Utilização
Inteiro (int)	56
Ponto flutuante (float)	6.18
Complexo (!)	2.6 + 8.4j
Booleanos (bool)	true/false
<i>String</i>	'casa'
Tupla	(1, 'aaa')
Lista	[1, 2.4, 'abc', [1, 'b']]
Dicionário	{'a':1, 2:'b', [1, 2]:[3, 4]}

Estruturas
de dados
em Python

Exemplos:

- Tipo String

```
7 texto1 = "Esse é o primeiro texto.\n"  
8 texto2 = 'Esse é o segundo texto.'  
9 print (texto1 + texto2)  
10 mensagem = '''  
11 Essa mensagem é uma  
12 'mensagem' que pode ser dividida  
13 em mais de uma "linha"  
14 '''  
15 print (mensagem)
```

- Tipo Booleano

```
7 condição = 4 > 6  
8 print (condição)  
9 condição = True  
10 print (condição)
```

Variáveis e Expressões

```
a = 3
b = 4
exp = (a + b)*4 - 35/5
print("O resultado da expressão eh igual a", exp)
```

Expressão aritmética

```
# a_str e b_str guardam strings
a_str = input("Digite o primeiro numero: ")
b_str = input("Digite o segundo numero: ")

# a_int e b_int guardam inteiros
a_int = int(a_str) # converte string/texto para inteiro
b_int = int(b_str) # converte string/texto para inteiro
```

Entrada de dados

Função de conversão int()

```
# calcule a soma entre valores que são números inteiros
soma = a_int + b_int
```

```
# imprima a soma
print("A soma de", a_int, "+", b_int, "eh igual a", soma)
```

Strings

- **Fatiamento** - Um substring de um string é chamado de **fatia** (do inglês slice).

```
▶ singers = "Peter, Paul, and Mary"  
print(singers[0:5])  
print(singers[7:11])  
print(singers[17:21])
```

```
☞ Peter  
Paul  
Mary
```

```
[7] fruta = "melancia"  
print(fruta[:3])  
print(fruta[3:])
```

```
mel  
ancia
```

```
[8] s = "python rocks"  
print(s[7:11]*3)
```

```
rockrockrock
```

- Uma **STRING** é definida como tudo que se encontra entre aspas simples (') ou duplas (").
- Existe uma terceira forma, que é a definição de strings **entre três aspas triplas**.

Strings

```
[18] print(ord("A"))  
      print(ord("B"))  
      print(ord("5"))  
      print(ord("a"))  
      print("apple" > "Apple")
```

65
66
53
97
True

```
▶ print(chr(65))  
  print(chr(66))  
  
  print(chr(49))  
  print(chr(53))  
  
  print("O caracter correspondente a 32 é",chr(32),"!!!")  
  print(ord(" "))
```

A
B
1
5
O caracter correspondente a 32 é !!!
32

Strings são imutáveis

```
▶ conversa = "Ola, mundo!"  
  conversa[0] = 'B' # ERROR!  
  print(conversa)
```

```
▶ conversa = "Ola, mundo!"  
  nova_conversa = 'B' + conversa[1:]  
  print(nova_conversa)  
  print(conversa)
```

Bla, mundo!
Ola, mundo!

```
1 string9 = "Olá, meu nome é Felipe"  
2 print(string9[0])  
3 print(string9[3])  
4 print(string9[21])
```

O
,
e

Refinando a formatação de saída

Strings literais formatadas



```
year = 2016  
event = 'Maratona'  
f'Eventos de {year} {event}'
```

```
↳ 'Eventos de 2016 Maratona'
```



```
import math  
  
print(f'O valor de pi é aproximadamente {math.pi:.5f}.')
```

Importar módulo
(biblioteca)

O valor de pi é aproximadamente 3.14159.


Desafio da aula anterior

1. Inscreva-se na plataforma Beecrowd: <https://judge.beecrowd.com/en/login>
2. Resolva os exercícios de números: 1073, 1095

beecrowd | 1095



Sequencia IJ 1

Adaptado por Neilor Tonin, URI  Brasil

Timelimit: 1

Você deve fazer um programa que apresente a sequencia conforme o exemplo abaixo.

Entrada

Não há nenhuma entrada neste problema.

Saída

Imprima a sequencia conforme exemplo abaixo

Exemplo de Entrada

Exemplo de Saída


```
I=1 J=60
I=4 J=55
I=7 J=50
...
I=? J=0
```


```
i=1
j=60
while j>=0:
    print(f"I={i} J={j}")
    i +=3
    j -= 5
```


Desafio da aula anterior

1. Inscreva-se na plataforma Beecrowd: <https://judge.beecrowd.com/en/login>
2. Resolva os exercícios de números: 1073, 1095

beecrowd | 1073

Quadrado de Pares
Adaptado por Neilor Tonin, URI  Brasil
Timelimit: 1



Leia um valor inteiro **N**. Apresente o quadrado de cada um dos valores pares, de 1 até **N**, inclusive **N**, se for o caso.

Entrada

A entrada contém um valor inteiro **N** ($5 < N < 2000$).

Saída

Imprima o quadrado de cada um dos valores pares, de 1 até **N**, conforme o exemplo abaixo.

Tome cuidado! Algumas linguagens tem por padrão apresentarem como saída $1e+006$ ao invés de 1000000 o que ocasionará resposta errada. Neste caso, configure a precisão adequadamente para que isso não ocorra.

Exemplo de Entrada	Exemplo de Saída
6	$2^2 = 4$ $4^2 = 16$ $6^2 = 36$

```
nro = int(input())
```

```
for i in range(1,nro+1):  
    if i%2==0:  
        print(f"{i}^2 = {i**2}")
```

Exercícios em Python

1. Analise o código, Python, abaixo e preencha a tabela referente ao Teste de Mesa do código em Python. Preencha também no Console o resultado a ser impresso.

CÓDIGO

```
n = 10
a, b = 0, 1
while a < n:
    print(a, end=' ')
    a, b = b, a+b
print()
```

TESTE DE MESA

n	a	b

CONSOLE

Exercícios em Python

2. Analise o código, Python, abaixo e preencha a tabela referente ao Teste de Mesa do código em Python. Preencha também no Console o resultado a ser impresso.

CÓDIGO

```
#definição da função
def fib(x):
    a, b = 0, 1
    while a < x:
        print(a, end=' ')
        a, b = b, a+b
    print()

#chamada da função fib() no PP
n = int(input())
fib(n)
```

TESTE DE MESA

n	x	a	b

CONSOLE

3. Analise as diferenças encontradas no código e nos testes.

Exercícios em Python

3. Implemente uma função `floyd()` que imprima o triângulo de Floyd para $n > 0$. Ex:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21

Strings e seus métodos

- `len(string)` - retorna o tamanho de caracteres da string
- `upper()` - retorna um string todo em maiúsculas
- `lower()` - retorna um string todo em minúsculas
- `capitalize()` - retorna um string com o primeiro caractere em maiúscula, e o resto em minúsculas
- `strip()` - retorna um string removendo caracteres em branco do início e do fim
- `lstrip()` - retorna um string removendo caracteres em branco do início
- `rstrip()` - retorna um string removendo caracteres em branco do fim
- `count(item)` - retorna o número de ocorrências de item
- `replace(old, new)` - substitui todas as ocorrências do substring old por new
- `center(largura)` - retorna um string centrado em um campo de tamanho largura
- `ljust(largura)` - retorna um string justificado à esquerda em um campo de tamanho largura
- `rjust(largura)` - retorna um string justificado à direita em um campo de tamanho largura
- `find(item)` - retorna o índice mais à esquerda onde o substring item é encontrado
- `rfind(item)` - retorna o índice mais à direita onde o substring item é encontrado
- `index(item)` - como find, mas causa um erro em tempo de execução caso item não seja encontrado
- `rindex(item)` - como rfind, mas causa um erro em tempo de execução caso item não seja encontrado

- Você deve experimentar esses métodos para que possa entender o que cada um faz. Observe que os métodos que retornam uma string não alteram a original.
- Dúvidas consulte as referências indicadas.

Exercícios em Python

4. Considere o código abaixo, em Python:

```
nome = input("Digite seu nome: ")
idade = int(input("Digite sua idade: "))
cidade = input("Digite o nome de sua cidade: ")

apresentacao = f"\nOlá! Meu nome é {nome}. Eu tenho {idade} anos e moro em {cidade}."
print(apresentacao)
```

Determine e imprima:

- quantos caracteres tem a string "apresentacao" para uma dada entrada de dados via teclado;
- troque o nome da cidade para São Paulo, caso seja diferente; e, imprima a string "apresentacao" novamente;
- Troque todas as vogais da string "apresentacao" por '*' e, imprima a mesma novamente.

Leituras para a próxima aula:

- Continue estudando a documentação da linguagem de programação Python.
- As três principais estruturas de dados utilizadas em ordenação e alocação de dados são as estruturas de dados dinâmicas: lista, fila e pilha. Essas estruturas são fundamentais para o desenvolvimento de aplicações; assim como as árvores. Veja o link:
<https://docs.python.org/pt-br/3/tutorial/datastructures.html>



Não se esqueçam do Uniube+