

Linguagem de Programação Orientada a Objetos I

Python – Introdução
Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

História

- ▶ Criada em 1989 por Guido Van Rossum, trabalhando hoje no Dropbox
- ▶ Origem do nome por homenagem ao seriado Monty Python
- ▶ Influências das linguagens ABC, Haskell, C, Perl, SmallTalk e Modula

Características

- ▶ Interpretada
- ▶ Portável (multi-plataforma)
- ▶ Extensível (C, Java, .Net)
- ▶ Livre
- ▶ Em Python, tudo é objeto
- ▶ Multiparadigma
 - Procedural
 - Orientada a Objetos
 - Funcional
- ▶ Case-sensitive

Características

- ▶ Simples e legível
- ▶ Suporte nativo a estruturas de dados de alto nível
- ▶ Sem declaração de variáveis
- ▶ Tipagem forte e dinâmica
- ▶ Controle de escopo por identação

Download e Instalação

- ▶ Site oficial:
 - <https://python.org/download>
- ▶ Versão usada:
 - Python 3.5
- ▶ Suporte de sintaxe da maioria dos editores

Tipos Primitivos

- ▶ int/long
- ▶ float
- ▶ bool – True/False
- ▶ Str – String
- ▶ Tuple – Registros
- ▶ list – Lista de objetos
- ▶ dict – Coleção chave/valor
- ▶ set – Coleção arbitrária

Tipos

- ▶ Quando uma variável é criada, seu tipo não pode mais ser alterado

```
>>> x = 1
>>> print(x)
1
>>> type(x)
<class 'int'>
>>> y = 2.5
>>> type(y)
<class 'float'>
>>> z = (int)y
      File "<stdin>", line 1
          z = (int)y
                  ^
SyntaxError: invalid syntax
>>> z = int(y)
>>> print(z)
2
>>> █
```

Variáveis

- ▶ Fortemente tipada
- ▶ Tipagem dinâmica
- ▶ Tudo é objeto
- ▶ Não é necessário declarar

Características dos tipos

- ▶ Tipo int englobe valores long também

```
[>>> x = 123561
[>>> type(x)
<class 'int'>
[>>> grande = x ** 12
[>>> print(grande)
12664177051358979477135584432535550802333736729296303230260321
[>>> type(grande)
<class 'int'>
>>> █
```

Características dos tipos

- ▶ Booleanos são True e False (notem o início com letra maiúscula)

```
>>> completo = False
>>> print(completo)
False
>>> completo = not completo
>>> print(completo)
True
>>> type(completo)
<class 'bool'>
>>> bool('')
False
>>> bool(0)
False
>>> bool(())
False
>>> bool([])
False
>>> bool({})
False
>>> bool(1)
True
>>>
```

Tudo é Objeto

- ▶ As variáveis podem ser iniciadas ou direto pelo valor ou com o uso de construtores, por exemplo:

```
>>> a = int('1111', 2)
>>> a
15
>>> type(a)
<class 'int'>
>>> █
```

Atribuição Mútua

```
[>>> nome = 'Tales'  
[>>> sobrenome = 'Viegas'  
[>>> print(nome, sobrenome)  
Tales Viegas  
[>>> nome,sobrenome = sobrenome,nome  
[>>> print(nome, sobrenome)  
Viegas Tales  
>>> █
```

Operadores Aritméticos

- ▶ Soma: $1 + 1$
- ▶ Subtração: $1 - 1$
- ▶ Multiplicação: $1 * 1$
- ▶ Divisão inteira: $5 / 2$
- ▶ Divisão ponto flutuante: $5 / 2.$
- ▶ Módulo: $10 \% 3$
- ▶ Exponenciação: $2 ** 4$

Operadores Comparação

- ▶ $2 > 1$
- ▶ $3 == 2$
- ▶ $5 != 10$
- ▶ $12 >= 12$
- ▶ $17 <= 15$
- ▶ $5 < x < 20$

Operadores Lógicos

- ▶ and
- ▶ or
- ▶ not

Strings

- ▶ Podem ser declaradas com simples ou duplas
- ▶ Slices

```
>>> texto = "Teste do Tales"
>>> texto[0]
'T'
>>> texto[12]
'e'
>>> texto[4:]
'e do Tales'
>>> texto[:3]
'Tes'
>>> texto[4:10]
'e do T'
>>> texto[1::2]
'te oTls'
>>> texto[1::3]
'eoas'
>>> █
```

```
...>>> texto[-1]
's'
>>> texto[-6]
' '
>>> texto[:-6]
'Teste do'
>>> texto[-4:]
'ales'
>>> texto[::]
'Teste do Tales'
>>> texto[::-1]
'selaT od etset'
```

Strings

▶ Alguns operadores para trabalhar com Strings

```
>>> nome = 'Tales' + ' ' + 'Viegas'  
>>> nome  
'Tales Viegas'  
>>> idade = 40  
>>> info = 'Nome: %s\nIdade: %d' % (nome, idade)  
>>> print(info)  
Nome: Tales Viegas  
Idade: 40  
>>> print('a' * 5)  
aaaaa  
>>> print('legal ' * 3)  
legal legal legal  
>>>
```

Strings

▶ Fugindo dos caracteres especiais

```
>>> print('c:\\pasta\\teste')
c:\\pasta      este
>>> print('c:\\\\pasta\\\\teste')
c:\\pasta\\teste
>>> print(r'c:\\pasta\\teste')
c:\\pasta\\teste
>>> █
```

Strings de múltiplas linhas

```
[>>> hinodogremio = """Até a pé nós iremos  
[... para o que der e vier  
[... mas o certo é que nós estaremos  
[... com o Grêmio onde o Grêmio estiver"""  
[>>> print(hinodogremio)  
Até a pé nós iremos  
para o que der e vier  
mas o certo é que nós estaremos  
com o Grêmio onde o Grêmio estiver  
... ■
```

Métodos de um objeto

► Comando dir(<tipo>)

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

Documentação Interativa

- ▶ Exemplos:

- `help(str)`
 - `help("".find)`

Entrada de Dados Console

- ▶ Comando input (retorna String)

```
>>> nome = input('Qual o seu nome: ')
Qual o seu nome: Tales
>>> nome
'Tales'
>>> idade = input('Qual a sua idade %s: ' % nome)
Qual a sua idade Tales: 40
>>> idade
'40'
```

Estruturas Condicionais

```
01_condicional.py  x

nomecompleto = input('Qual o seu nome completo: ')
nome, sobrenome = nomecompleto.split(" ", 1)
idade = int(input('Qual a sua idade %s: ' % nome))
localfamilia = input('De onde é a sua família %s: ' % nome)

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

```
[MacBook-Pro-de-Tales:AulaPython talesviegas$ python3 01_condicional.py
Qual o seu nome completo: Tales Bitelo Viegas
Qual a sua idade Tales: 40
De onde é a sua família Tales: Cachoeirinha
Você é maior de idade.
MacBook-Pro-de-Tales:AulaPython talesviegas$ ]
```

Blocos de Comandos

- ▶ Python não usa { e } (ou begin/end) para delimitar os blocos de comandos
- ▶ O indicador de início de bloco de comando é o character : (dois pontos)
- ▶ O bloco é identificado pela identação (por padrão 4 espaços)
- ▶ Utilitário pycodestyle verifica os estilos de código em Python

Estruturas Condicionais

- ▶ Não existe switch
- ▶ Pode ser usado o elif

```
>>> calc = eval(input("Digite uma operação: "))
Digite uma operação: 4+5
>>> if (calc > 5) :
...     print("calc maior que 5")
... elif (calc < 5) :
...     print("calc menor que 5")
... else :
...     print("calc igual a 5")
...
calc maior que 5
>>> █
```

Estruturas Condicionais

- ▶ Condicional de uma linha

```
print("Seu nome é", "maior" if (len(nome) >  
len("Tales")) else "menor", "que o meu")
```

Estruturas de Repetição

- ▶ for e while
- ▶ Possui break e continue
- ▶ Possuem else, que é executado se não houve break no código

```
>>> i = 1
>>> while i < 100:
...     print(i)
...     i+=1
...
1
2
3
4
5
>>> empresa = "Ulbra"
>>> for letra in empresa:
...     print(letro)
...
U
l
b
r
a
>>> for i in range(50,100,2):
...     print(i)
...
50
52
54
```