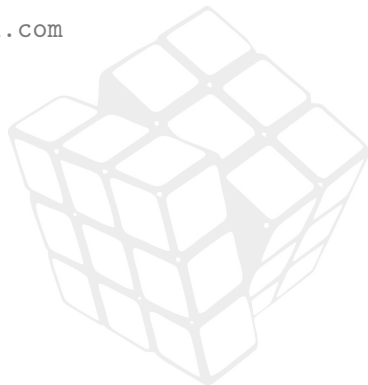


Algoritmos e Programação

Introdução a Python

Gustavo Sávio
gsoprofessor@gmail.com

2017.2



É uma linguagem de programação de **alto nível** criada em 1991 por Guido Van Rossum

- ▶ O principal objetivo: Produtividade e legibilidade
- ▶ Código simples de escrever e dar manutenção

Características

- ▶ Baixo uso de caracteres especiais
- ▶ Uso de indentação para definir blocos de código
- ▶ Múltiplos paradigmas de programação
- ▶ Uma vasta biblioteca padrão
- ▶ Linguagem livre (*open source*)
- ▶ Multiplataforma
- ▶ Interpretada

Python Software Foundation



- ▶ *<http://www.python.org>*

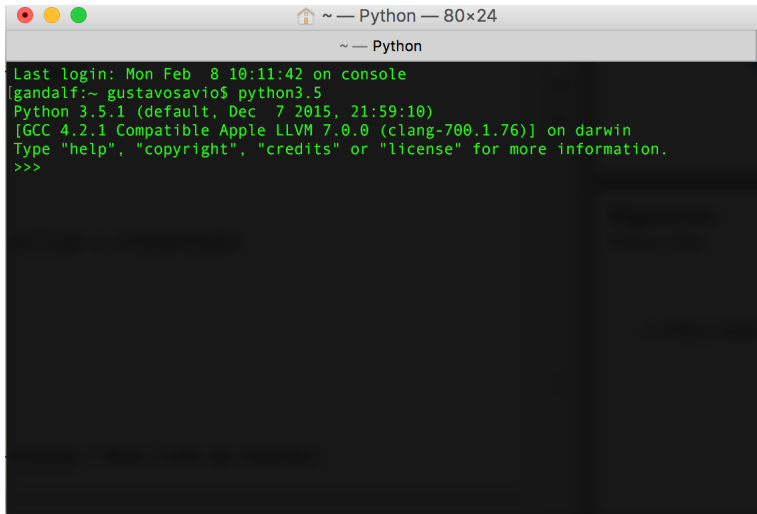


Python é interpretado...

- ▶ Existem duas maneiras de utilizar o interpretador
- ▶ O modo linha de comando (Iterativo)
- ▶ O modo *script*

Algoritmo

Python / Modo linha de comando / Iterativo

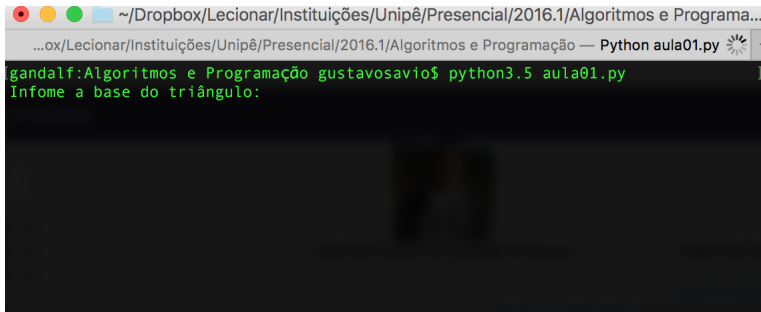


```
~ — Python — 80x24
~ — Python

Last login: Mon Feb  8 10:11:42 on console
[gandalf:~ gustavosavio$ python3.5
Python 3.5.1 (default, Dec  7 2015, 21:59:10)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Algoritmo

Python / Modo de Script



```
~/Dropbox/Lecionar/Instituições/Unipê/Presencial/2016.1/Algoritmos e Programa...  
...ox/Lecionar/Instituições/Unipê/Presencial/2016.1/Algoritmos e Programação — Python aula01.py  
gandalf:Algoritmos e Programação gustavosavio$ python3.5 aula01.py  
Informe a base do triângulo:
```


Algoritmo

Python / Função print()

```
1 print("Hello World!")
2 print(1 + 1)
```

```
1 # Exemplo 1
2 print("%.2f" % (10.0000))
3 print("%.2f %.2f" % (10.0000, 1222.9999))
4
5
6 # Exemplo 2 com a função format
7 print("{valor:.2f}".format(valor=10.0000))
8 print("{valor:.2f} {valor2:.2f}".format(valor=12.0000,
    ↪ valor2=12.988))
```

No processo de desenvolvimento, constantemente iremos manipular valores

- ▶ Por exemplo: 1, "valor", 1.89
- ▶ Cada valor possui um *tipo de dado* definido
- ▶ Cada tipo de dado é processado de um modo
- ▶ Ex: Números inteiros não são processados como uma cadeia de caracteres (**string**)

Onde iremos armazenar esses valores?

- ▶ Na memória
- ▶ Variáveis são nomes dados a áreas da memória
- ▶ Essas áreas são utilizadas para guardar valores
- ▶ Ou seja, é um nome que se refere a um valor
- ▶ Podemos definir o valor de uma variável com o operador de atribuição =

```
1 professor = "Gustavo"  
2 disciplina = "Algoritmos e Programação"  
3 periodo = 1  
4  
5 print(professor, disciplina, periodo)
```

► nomeVariável = expressão

```
1 professor, disciplina, periodo = "Gustavo", "Algoritmos e  
  ↪  Programação", 1  
2  
3 print(professor, disciplina, periodo)
```

► nomeVariável1,nomeVariável1, N = expressão1, expressão2, N ...

Existem regras para definirmos nomes de variáveis

- ▶ Nomes significativos
- ▶ Podem conter letras e números
- ▶ Tem que começar com uma letra
- ▶ Por convenção devem ser minúsculas
- ▶ Minúsculas e Maiúsculas são diferentes: **nome** é diferente de **Nome**
- ▶ Podemos utilizar o caracteres **underline** exemplo:
nome_muito_longo
- ▶ Não podem utilizar as palavras reservadas da linguagem

```
1 import sys
2 import keyword
3
4
5 print("Python version: ", sys.version_info)
6 print("Python keywords: ", keyword.kwlist)
```

- Lista de palavras reservadas

```
1  ''' Identificadores válidos '''
2  _salario = 7.500
3  salario = 7.500
4
5  ''' Identificadores inválidos '''
6  28_idade = 28
7  int = "Classe"
```

- ▶ As variáveis são criadas dinamicamente
- ▶ São dinamicamente tipadas conforme o valor atribuído

```
1  _area_km = 56.585
2  estado = "paraíba"
3  brasil = True
4
5  print(type(_area_km))
6  print(type(estado))
7  print(type(brasil))
```

- ▶ Numéricos
- ▶ **int**: 1, 10, 27 (int e long unificado no Python 3)
- ▶ **float**: 10.8, 89.7
- ▶ **complex**: 10J. Exemplo $1 + 2j * 3$
- ▶ **string** (Cadeia de caracteres): Representadas por aspas simples ou duplas e utiliza o operador `+` para concatenar cadeias de caracteres
- ▶ **bool**: Representação booleana - True or False

- ▶ Soma: $1 + 2$
- ▶ Subtração: $5 - 3$
- ▶ Multiplicação: $10 * 2$
- ▶ Potência $2 ** 2$
- ▶ Divisão: $10 / 2$
- ▶ Divisão descartando parte fracionária: $9 // 2$
- ▶ Resto de divisão: $10 \% 3$

- ▶ Python é tipado dinamicamente
- ▶
- ▶ Programas podem processar números de diferentes tipos
- ▶ $1 + 2.22 = \text{float}$
- ▶ $\text{True} + 1 = \text{int}$
- ▶ $"10" + 10$

```
1  Traceback (most recent call last):  
2      File "<stdin>", line 1, in <module>  
3  TypeError: Can't convert 'int' object to str implicitly
```

Mas se tivéssemos certeza que a string representaria um número?
O que poderíamos fazer?

```
1  # castings válidos
2
3  int("10") + 10
4  float("2.88") + 10
5  bool(0)
6  bool(1)
7
8  # castings inválidos
9
10 int("True") + 10
11 float("nome do usuário") + 10
```

```
1  # coding: utf-8
2
3  __author__="Gustavo Sávio"
4
5  nome = input("Informe o seu nome: ")
6  altura = float(input("Informe a sua altura: "))
7  peso = float(input("Informe o seu peso: "))
8  imc = peso / altura**2
9
10 print("O seu IMC é: ", imc)
```

```
1  # coding: utf-8
2
3  __author__="Gustavo Sávio"
4
5  nome = input("Informe o seu nome: ")
6  notas = input("Informe as suas notas separadas por vírgulas ")
7  nota1, nota2 = notas.split(",")
8
9  print("A sua média é: ", (float(nota1) + float(nota2)) / 2)
```

```
1 # coding: utf-8
2
3 __author__="Gustavo Sávio"
4
5 nome = input("Informe o seu nome: ")
6 notas = input("Informe as suas notas separadas por vírgulas ")
7 nota1, nota2 = notas.split(",")
8
9 print("A sua média é: ", float(nota1) + float(nota2) / 2)
```

► Qual é o resultado?




- ▶ 1. (): Tudo que estiver dentro dos parênteses mais internos
- ▶ 2. **
- ▶ 3. *, /, %
- ▶ 4. +, -

```
1  ( 10 + 2) ** 2 * (9 - 4) + 19
2      12 ** 2 * 5 + 19
3      144 * 5 + 19
4      720 + 19
5      739
6
7  print(( 10 + 2) ** 2 * (9 - 4) + 19)
```

Associatividade é a ordem no qual uma expressão composta por operadores com a mesma precedência serão avaliados

- ▶ Existem operadores com o mesmo nível de precedência, por exemplo multiplicação e divisão
- ▶ A maioria dos operadores no Python possuem associatividade da esquerda para direita

```
1 # Associatividade da esquerda para direita
2
3 5 * 2 // 3 # 3
4 5 * (2 // 3) # 0
5
6 # Associatividade da direita pra esquerda
7
8 2 ** 3 ** 2 # 512
9 (2 ** 3) ** 2 # 64
```

-  Python Software Foundation. <https://www.python.org/psf/>
-  Pyscience Brasil. <http://pyscience-brasil.wikidot.com>
-  Allen B. Downey; Think Python How to Think Like a Computer Scientist.