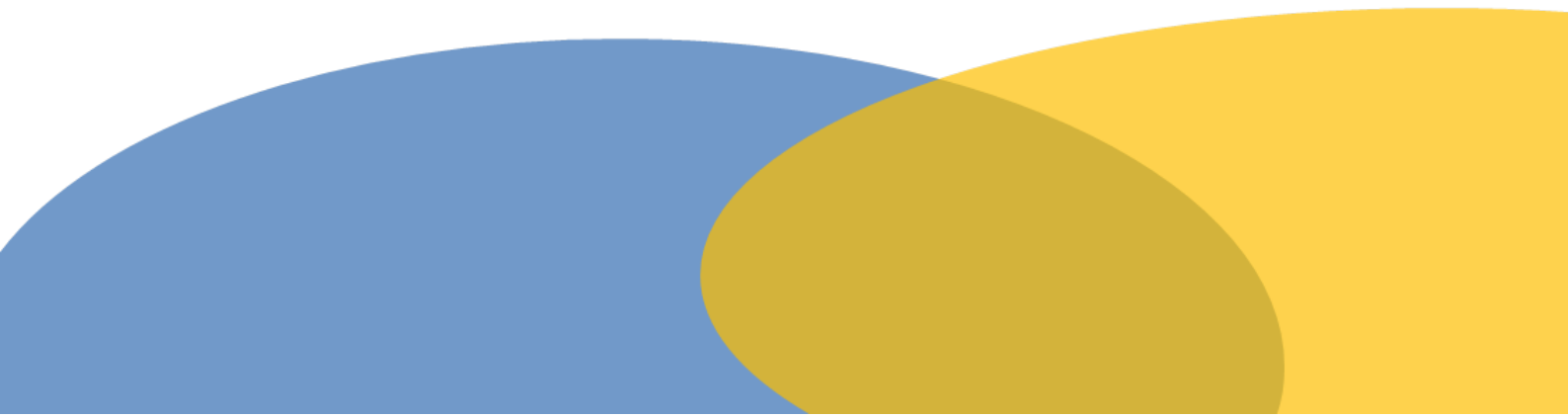
Introdução à Linguagem Python

Paradigmas de Linguagens de Programação

Alysson de Jesus Alcantara Alves

Ausberto S. Castro Vera

29 de agosto de 2021

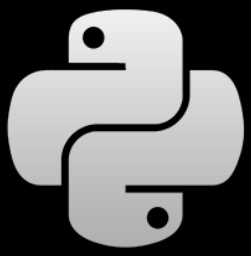


UENF - UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CCT - CENTRO DE CIÊNCIA E TECNOLOGIA
LCMAT - LABORATÓRIO DE MATEMÁTICAS
CC - CURSO DE CIÊNCIA DA COMPUTAÇÃO

Primeira edição, Maio 2019

Sumário



```
print “Hello World”
```

1. Introdução

> "As pessoas ainda são loucas pelo Python mesmo depois de vinte e cinco anos, o que é difícil de acreditar"—Michael Palin

Concebido originalmente no final dos anos 80, por por Guido van Rossum no Centrum Wiskunde & Informatica (CWI) na Holanda, veio como sucessor da linguagem ABC, a implementação se deu inicio em dezembro de 1989. Python foi lançado como sendo uma linguagem de propósito geral de alto nível, multiparadigma, suportando paradigmas orientados a objetos, imperativos, funcionais e procedurais, com uma tipagem dinâmica e de fácil leitura, característica por exigir poucas linhas se comparado a outras linguagens, por conta disso Python é tido como uma das linguagens mais populares da atualidade, considerado a terceira linguagem "mais amada"de acordo com uma pesquisa conduzida pelo site do Stack Overflow em 2018 e está entre as 5 linguagens mais populares, de acordo com uma pesquisa conduzida pela RedMonk

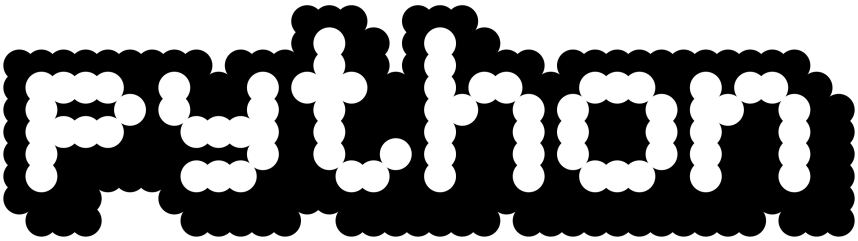
O nome Python teve a sua origem no grupo humorístico britânico Monty Python,[8] criador do programa Monty Python's Flying Circus, embora muitas pessoas façam associação com o réptil do mesmo nome (em português, píton ou pitão).

> Em fevereiro de 1991, Van Rossum publicou o código (rotulado como versão 0.9.0) no grupo de discussão da alt.sources. Nessa versão já estavam presentes classes com herança, tratamento de exceções, funções e os tipos de dado nativos list, dict, str, e assim por diante. Também estava presente nessa versão um sistema de módulos emprestado do Modula-3. O modelo de exceções também lembrava muito o do Modula-3, com a adição da opção else clause. Em 1994 foi formado o principal fórum de discussão do Python, comp.lang.python, um marco para o crescimento da base de usuários da linguagem. [Ven03]

1.1 Aspectos históricos da linguagem Python

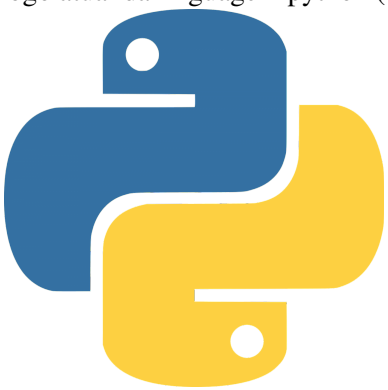
Python chegou a versão 1.0 em janeiro de 1994. As principais novas funcionalidades incluídas nesta versão foram as ferramentas de programação funcional lambda, map, filtere reduce. Van Rossum afirmou que "Python adquiriu lambda, reduz (), filtro () e mapa (), cortesia de um hacker Lisp que os perdeu e enviou patches de trabalho ". [vvR05] A última versão lançada enquanto Van Rossum estava no CWI foi o Python 1.2. Em 1995, Van Rossum continuou seu trabalho em Python na Corporation for National Research Initiatives (CNRI) em Reston , Virgínia , de onde lançou várias versões. Abaixo pode-se ver a evolução das logos do Python

Figura 1.1: Primeira logo da linguagem Python (1990s–2006)



Fonte: domínio público

Figura 1.2: Logo atual da linguagem python (2006 - atual)



Fonte: community logo python

1.2 Áreas de Aplicação da Linguagem

Por ter uma sintaxe simples e ser baseado em c, Python é uma linguagem que pode ser facilmente escalável e usada em diversas áreas diferentes.

Python pode servir como uma linguagem de script para aplicativos da web, por exemplo, via mod_wsgi para o servidor da web Apache Com a Web Server Gateway Interface, uma API padrão evoluiu para facilitar essas aplicações. Frameworks da Web como Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle e Zope apoiam os desenvolvedores no design e manutenção de aplicações complexas. Pyjs e IronPythonpode ser usado para desenvolver o lado do cliente de aplicativos baseados em Ajax. SQLAlchemy pode ser usado como um mapeador de dados para um banco de dados relacional. Twisted é um framework para programar comunicações entre computadores e é usado (por exemplo) pelo Dropbox [Res15]

1.2.1 Big Data

Python, possui bibliotecas voltada diretamente Big Data, como a Scikit-learn, que oferece funcionalidades para o pré-processamento, a classificação e a clusterização dos dados, a seleção de modelos e a aplicação de algoritmos de regressão. Das várias bibliotecas que o python possui para realizar cálculo de Big Data podemos citar a Imbalanced-learn, Jupyter, Pandas e Scikit-learn. No artigo de Ivan Eduardo o mesmo realiza uma série de aplicações de Big Data utilizando cada uma das bibliotecas [Kü15]

1.2.2 Orientação a objetos

[Woi03] A programação orientada a objetos (OOP) é um paradigma de programação baseado no conceito de "objetos ", que podem conter dados e código: dados na forma de campos (geralmente conhecidos como atributos ou propriedades), e código, na forma de procedimentos (freqüentemente conhecido como métodos).

Em Python, todo valor é na verdade um objeto. Seja uma tartaruga, uma lista, ou mesmo um inteiro, todos são objetos. Programas manipulam esses objetos realizando computações diretamente com eles ou chamando os seus métodos (ou seja, pedindo que esses objetos executem seus métodos). Para ser mais específico, nós dizemos que um objeto possui um estado e uma coleção de métodos que ele pode executar. O estado de um objeto representa as coisas que o objeto sabe sobre si mesmo. Por exemplo, como vimos com os objetos tartaruga, cada tartaruga possui um estado que representa a sua posição, sua cor, sua direção, etc. Cada tartaruga também tem a capacidade de se mover para a frente, para trás, ou virar para a direita ou esquerda. Cada tartaruga é diferente pois, embora sejam todas tartarugas, cada uma tem um estado diferente (como posições diferentes, ou orientações, etc).

Sabendo como eles funcionam podemos ver um exemplo prático na linguagem python criando um objeto ponto que possui o eixo x e o eixo y

Figura 1.3: Código de OOP em python, criando um ponto com dois eixos

```
1  class Point:
2      """ Point class for representing and manipulating x,y coordinates. """
3
4      def __init__(self):
5          """ Create a new point at the origin """
6          self.x = 0
7          self.y = 0
```

Fonte: Autor

1.2.3 Data Science

[Gru16] Pode-se dizer que um cientista de dados, são para propósitos práticos, estatísticos que entendem sobre ciência da computação, alguns são mais estatísticos e outros são mais engenheiros de software, mas ambos os perfis são cientistas de dados. A Ciência de dados serve para extrai todos os dados que estão bagunçados dentro da sociedade, e transforma-los em conhecimento, que servem como guia para melhorar nossa sociedade.

No exemplo abaixo é criado um histograma para agrupar dados unificados em agrupamentos (buckets) discretos e contar quantos pontos vão para cada um, no exemplo esses dados podem representar varias situações como a media diária em minutos que cada usuário passa no seu site.

Trecho de código em python para definir o histograma

```
def bucketize(point, bucket_size):
"""reduza o ponto para o pr ximo"""
"""m ltiplo mais baixo de bucket_size"""
return
    bucket_size * math.floor(point / bucket_size)

def make_histogram(points, bucket_size):
"""agrupa os pontos e conta quantos em cada bucket"""
return
    Counter(bucketize(point, bucket_size) for point in points)

def plot_histogram(points, bucket_size, title=""):
    histogram = make_histogram(points, bucket_size)
    plt.bar
     (histogram.keys(), histogram.values(), width=bucket_size)
    plt.title(title)
    plt.show()
```

Agora consideramos os seguintes conjuntos de dados

```
random.seed(0)
# uniforme entre   100   e 100

uniform = [200 * random.random() - 100 for _ in range(10000)]
# distribui  o normal com m dia 0, desvio padr o 57

normal = [57 * inverse_normal_cdf(random.random())
for _ in range(10000)]
```

Ambos possuem médias próximas a 0 e desvios padrões próximos a 58. No entanto, possuem distribuições bem diferentes. A Figura 10-1 mostra a distribuição de uniform:

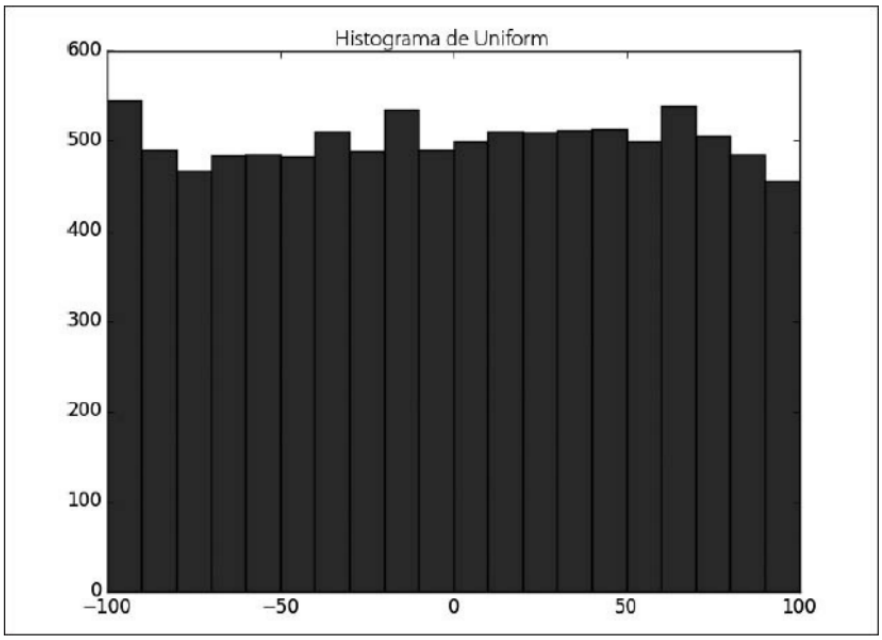
```
plot_histogram(uniform, 10, "Histograma de Uniform")
```

Já o trecho abaixo mostra a distribuição normal

```
plot_histogram(normal, 10, "Histograma Normal")
```

Nesse exemplo as duas possuem valores max e min que podem ser vistos no gráfico abaixo:

Figura 1.4: Histograma uniforme



Fonte: Livro - Data Science do zero: Primeiras regras com o Python

2. Conceitos básicos da Linguagem Python

Os livros básicos para o estudo da Linguagem Python são: [Sum13], [Gut15], [Per16]

Neste capítulo é apresentado ....

Segundo [Seb11], a linguagem Python, . . .

De acordo com [Seb11] e [RH04], a linguagem Python . . .

[Seb11] afirma que a linguagem Python . . .

Considerando que a linguagem Python ([Seb11], [Wat90]) é considerada como ....

2.1 Variáveis e constantes

2.2 Tipos de Dados Básicos

2.2.1 String

Um string é uma sequência de caracteres considerado como um item de dado simples. Para Python, um string é um array de caracteres ou qualquer grupo de caracteres escritos entre doble aspas ou aspas simples. Por exemplo,

```
>>> #usando aspas simples
>>> pyStr1 = 'Brasil'
>>> print (pyStr1)  Brasil
>>> #usando aspas duplas
>>> pyStr2 = "Oi, tudo bem?"
>>> print (pyStr2)
Oi, tudo bem?
```

- *Concatenação de strings*
  Strings podem ser concatenadas utilizando o operador +, e o seu comprimento pode ser calculado utilizado o operador `len(string)`

```
>>> # concatenando 2 strings
>>> pyStr = "Brasil" + " verde amarelo"
>>> print (pyStr)
```

```
    Brasil verde amarelo
    >>> print (len(pyStr))
    20
```

- *Operador de indexação*
  Qualquer caracter de um string o sequência de caracteres pode ser obtido utilizando o operador de indexação []. Existem duas formas de indexar em Python, os caracteres de um string:

  Index com inteiros positivos indexando a partir da esquerda começando com 0 e onde 0 é o index do primeiro caracter da sequência

  Index com inteiros negativos indexando a partir da direita começando com -1, e onde -1 é o último elemento da sequência, -2 é o penúltimo elemento da sequência, e assim sucessivamente.

```
    >>> # Indexando strings
    >>> pyStr = "Programando"
    >>> print (len(pyStr))
    11
    >>> print (pyStr)
    Brasil verde amarelo
```

- *Operador de Fatias*
  O operador de acesso a itens (caracteres individuais) também pode ser utilizado como operador de fatias, para extrair uma fatia inteira (subsequência) de caracteres de um string. O operador de Fatias possui três sintaxes:
  `seq[ inicio ]`
  `seq[ inicio : fim ]`
  `seq[ início : fim : step ]`
  onde `início`, `fim` e `step` são números inteiros.

```
    >>> # Indexando strings
    >>> pyStr = "Programando Python"
    >>> print (len(pyStr))
    11
    >>> print (pyStr)
    Brasil verde amarelo
```

2.3 Tipos de Dados de Coleção

2.3.1 Tipos Sequenciais

2.3.2 Tipos Conjunto

2.3.3 Tipos Mapeamento

2.4 Estrutura de Controle e Funções

2.4.1 O comando IF

2.4.2 Laço FOR

2.4.3 Laço WHILE

2.5 Módulos e pacotes

2.5.1 Módulos

2.5.2 Pacotes

Código fonte para a linguagem Python:

```
number_1 = int(input('Ingresse o primeiro numero: '))
number_2 = int(input('Ingresse o segundo numero: '))

# Soma
print('{} + {} = '.format(number_1, number_2))
print(number_1 + number_2)

# Substra\c{c}\~{a}o
print('{} - {} = '.format(number_1, number_2))
print(number_1 - number_2)

# Multiplica\c{c}\~{a}o
print('{} * {} = '.format(number_1, number_2))
print(number_1 * number_2)

# Divis\~{a}o
print('{} / {} = '.format(number_1, number_2))
print(number_1 / number_2)
```

3. Programação Orientada a Objetos com Python

3.1 Classes e Objetos

```
class NomeClasse:

def metodo1:

def Metodo2:
```

3.2 Operadores ou Métodos

3.3 Herança

3.4 Estudo de Caso:

4. Aplicações da Linguagem Python

Devem ser mostradas pelo menos CINCO aplicações completas da linguagem, e em cada caso deve ser apresentado:

- Uma breve descrição da aplicação
- O código completo da aplicação,
- Imagens do código fonte no compilador-interpretador,
- Imagens dos resultados após a compilação-interpretação do código fonte
- Links e referencias bibliográficas de onde foi obtido a aplicação

4.1 Operações básicas

4.2 Programas gráficos

4.3 Programas com Objetos

4.4 O algoritmo Quicksort - Implementação

4.5 Aplicações com Banco de Dados

5. Ferramentas existentes e utilizadas

Neste capítulo devem ser apresentadas pelo menos DUAS (e no máximo 5) ferramentas consultadas e utilizadas para realizar o trabalho, e usar nas aplicações. Considere em cada caso:

- Nome da ferramenta (compilador-interpretador)
- Endereço na Internet
- Versão atual e utilizada
- Descrição simples (máx 2 parágrafos)
- Telas capturadas da ferramenta
- Outras informações

5.1 Editor MNOP

5.2 Compilador XYZ

5.3 Interpretador UVW

5.4 Ambientes de Programação IDE MNP

6. Conclusões

Os problemas enfrentados neste trabalho ...

O trabalho que foi desenvolvido em forma resumida ...

Aspectos não considerados que poderiam ser estudados ou úteis para ...

Figura 6.1: Linguagens de programação modernas

Fonte: O autor

Referências Bibliográficas

[Gru16] Joel Grus. *Data Science do zero: Primeiras regras com o Python*. Alta Books, Rua Viúva Cláudio, 291 - Bairro Industrial do Jacaré, edição revista e ampliada edition, 2016. Citado na página 8.

[Gut15] John V. Guttag. *Introdução à Computação e Programação Usando Python*. Infopress Nova Mídia, São Paulo, edição revista e ampliada edition, 2015. Citado na página 11.

[Kü15] Ivan Eduardo Metz Kühne. Aplicação de técnicas de big data analytics às smart grids como forma de descoberta de padrões relevantes. *Universidade Regional do Noroeste do Estado do Rio Grande do Sul*, 2015. Citado na página 7.

[Per16] Ljubomir Perkovic. *Introdução à Computação usando Python: un foco no desenvolvimento de Aplicações*. LTC Livros Técnicos e Científicos Editora Ltda, Rio de Janeiro, 2016. Citado na página 11.

[Res15] Google Research. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *White Paper*, 2015. Citado na página 7.

[RH04] Peter Van Roy and Seif Haridi. *Concepts, Techniques and Models of Computer Programming*. The MIT Press, Cambridge, 2004. Citado na página 11.

[Seb11] Robert W. Sebesta. *Conceitos de Linguagens de Programação*. Bookman, Porto Alegre, 2011. Citado na página 11.

[Sum13] Mark Summerfield. *Programação em Python 3 - Uma Introdução Completa à Linguagem Python*. Biblioteca do Programador. Alta Books Editora, Rio de Janeiro, 2013. Citado na página 11.

[Ven03] Bill Venners. The making of python a conversation with guido van rossum, part i. *Commun. ACM*, 2003. Citado na página 5.

[vvR05] Guido van van Rossum. The fate of reduce() in python 3000. *Commun. ACM*, 2005. Citado na página 6.

[Wat90] David Anthony Watt. *Programming Language Concepts and Paradigms*. Prentice Hall International, London, 1990. Citado na página 11.

[Woi03] Josué Labaki & E.R Woiski. Python orientado a objetos. *Grupo Python, UNESP-Ilha Solteira*, 2003. Citado na página 7.

Disciplina: *Paradigmas de Linguagens de Programação 2019*
Linguagem: *Linguagem Python*
Aluno: *Nome Completo do aluno*

Ficha de avaliação:

Aspectos de avaliação (requisitos mínimos)	**Pontos**
Elementos básicos da linguagem (Máximo: 01 pontos) • Sintaxe (variáveis, constantes, comandos, operações, etc.) • Usos e áreas de Aplicação da Linguagem	
Cada elemento da linguagem (definição) com exemplos (Máximo: 02 pontos) • Exemplos com fonte diferenciada (Courier , 10 pts, azul)	
Mínimo 5 exemplos completos - Aplicações (Máximo : 2 pontos) • Uso de rotinas-funções-procedimentos, E/S formatadas • Menu de operações, programas gráficos, matrizes, aplicações	
Ferramentas (compiladores, interpretadores, etc.) (Máximo : 2 pontos) • Ferramentas utilizadas nos exemplos: pelo menos DUAS • Descrição de Ferramentas existentes: máximo 5 • Mostrar as telas dos exemplos junto ao compilador-interpretador • Mostrar as telas dos resultados obtidos nas ferramentas • Descrição das ferramentas (autor, versão, homepage, tipo, etc.)	
Organização do trabalho (Máximo: 01 ponto) • Conteúdo, Historia, Seções, gráficos, exemplos, conclusões, bibliografia	
Uso de Bibliografia (Máximo: 01 ponto) • Livros: pelo menos 3 • Artigos científicos: pelo menos 3 (IEEE Xplore, ACM Library) • Todas as Referências dentro do texto, tipo [ABC 04] • Evite Referências da Internet	
Conceito do Professor (Opcional: 01 ponto)	
Nota Final do trabalho:	

Observação: Requisitos mínimos significa a *metade* dos pontos