



Phyton

Projeto de Linguagem de Programação

Histórico

Foi criada por Guido van Rossum

O desenvolvimento inicial de Python
começou em um instituto de pesquisa em
Amsterdam chamado CWI

ABC linguagem usada pelo grupo CWI. Tinha
pontos negativos

Guido van Rossum

F

- Ditador Benevolente Vitalício da Linguagem Python



Nasceu em 31 de janeiro de 1956;
País: Holanda/Países Baixos
Diplomado pela Universidade de
Amsterdã em 1982;

Contexto para a criação




- Ocorreu na mesma época em que várias outras linguagens de programação dinâmicas
 - Tcl
 - Perl
 - Ruby
- Após o fracasso do projeto do grupo ABC, Guido passa a trabalhar no grupo Amoeba
- Phyton foi criado
 - Havia a necessidade de uma linguagem que "preencheria o vazio entre C e o shell". Por um tempo longo, esse foi o principal objetivo do Python.

Versões

G

Data de lançamento	Versão
Dezembro de 1989	Início da implementação
1990	Lançamento interno ao CWI (N. T.: <u>Centro de Matemática e Ciência da Computação</u>)
20 de fevereiro de 1991	0.9.0 (lançado em alt.sources)
Fevereiro de 1991	0.9.1
Outono de 1991	0.9.2
24 de dezembro de 1991	0.9.4
02 de janeiro de 1992	0.9.5 (somente para Macintosh)
06 de abril de 1992	0.9.6
1992 (mês desconhecido)	0.9.7 beta
09 de janeiro de 1993	0.9.8
29 de julho de 1993	0.9.9
26 de janeiro de 1994	1.0.0



Data de lançamento	Versão
05 de setembro de 2000	<u>1.6</u>
16 de outubro de 2000	<u>2.0</u>
17 de abril de 2001	<u>2.1</u>
21 de dezembro de 2001	<u>2.2</u>
29 de julho de 2003	<u>2.3</u>
30 de novembro de 2004	<u>2.4</u>
16 de setembro de 2006	<u>2.5</u>
01 de outubro de 2008	<u>2.6</u>
03 de dezembro de 2008	<u>3.0</u>

Existem versões mais atuais, sendo a 3.5.0 a mais nova.

Mudanças na comunidade python

- 1991- A mudança mais significativa seria a introdução de classes para a linguagem.
- 1994 - Primeira workshop
- 1995 - A comunidade Python começou a crescer e a importância dessa interação entre usuários e criadores foi o que gerou a linguagem como ela é vista hoje.
- 1996 até os dias atuais a python continua seu desenvolvimento com um comunidade forte e ativa.

Mudanças significativas

- Python 1.6.0 para 2.0.0 –
 - Existem várias melhorias importantes de sintaxe , descritos em mais detalhe abaixo :
 - Atribuição Aumentada , por exemplo, `x += 1`
 - Lista compreensões , por exemplo, `[x ** 2 para x no intervalo (10)]`
 - Extensão declaração de importação , por exemplo, `Import Module as Name`
 - Extensão declaração de impressão , por exemplo, `print>> file , “Hello “`
 - Outras mudanças importantes : Opcional coleção de lixo cíclica.

Mudanças significativas

- Python 2.x para 3.x
 - Lista de características presentes apenas na versão 3.x e não serão portadas para as versões 2.x:
 - strings are Unicode by default
 - clean Unicode/bytes separation
 - exception chaining
 - function annotations
 - syntax for keyword-only arguments
 - extended tuple unpacking
 - non-local variable declarations
- Guido van Rossum decidiu limpar Python 2.x corretamente, com menos consideração para compatibilidade com versões anteriores.

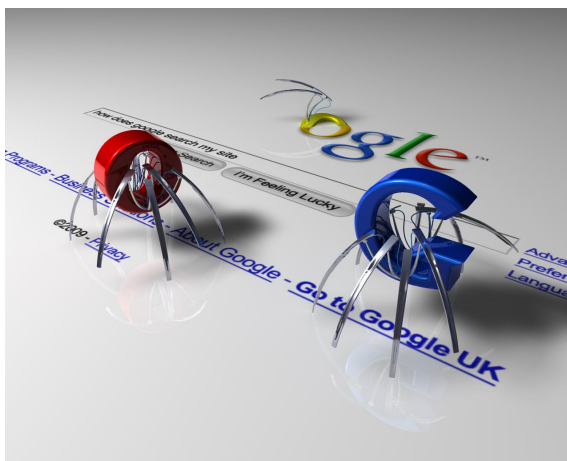
Premissas

- **Portabilidade:** códigos transferidos com pouca ou nenhuma modificação.
- **Extensão de módulos:** facilmente estendido com novas funções e tipos de dados implementados em C ou C++
- **Código Aberto:** comunidade grande e ativa.
- **Produtividade:** criação rápida e fácil de programas.

Usuário Característico

M

- Aplicações WEB



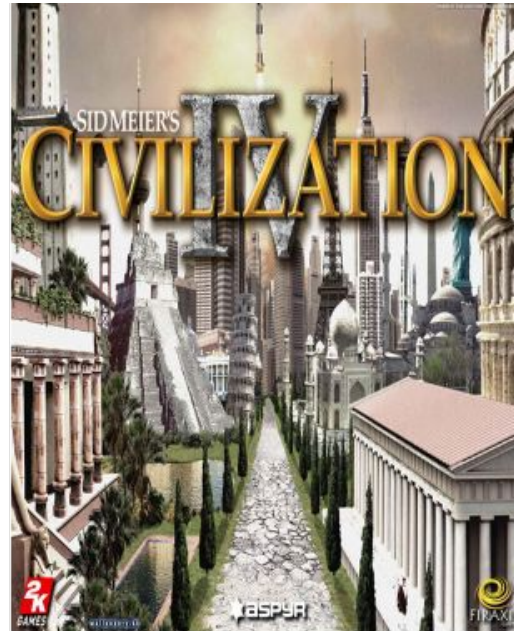
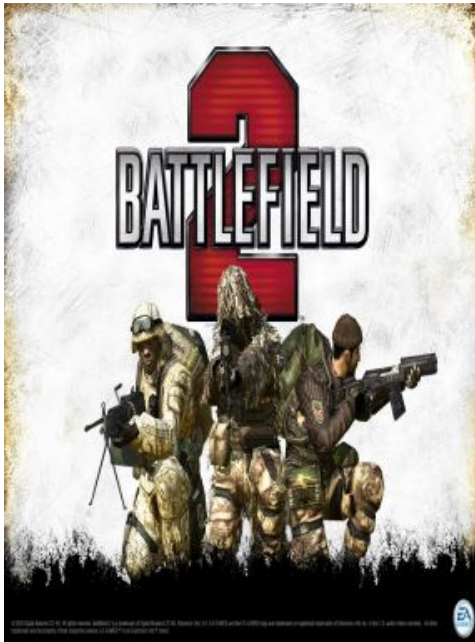
YouTube



Usuário Característico

M

- Jogos



Usuário Característico

M

- Gráficos



Usuário Característico

M

- Ciência
- Desenvolvimento de Software
- Governo



NOKIA



Domínio da Aplicação



- Desenvolvimento WEB e INTERNET
- Científico e Numérico
- Educação
- Desktop GUIs
- Desenvolvimento de software

Domínio da Aplicação

- Desenvolvimento WEB e INTERNET
 - - Alternativas
 - - Frameworks como Django e Pyramid.
 - - Micro-frameworks como Flask e Bottle.
 - - Gerenciamento de conteúdo, Plone e CMS.
 - - Protocolos suportados e bibliotecas
 - - HTML, XML, JSON, E-Mail, FTP, IMAP, outros.
 - - Requests, BeautifulSoup, Feedparser, outros.
- Científico e Numérico
 - - SciPy .: matemática, ciência e engenharia.
 - - Panda .: análise de dados e modelagem
 - - Ipython.: edição e registro de sessões de trabalho, programação paralela.

Domínio da Aplicação

- Educação
 - - Excelente para o ensino de programação
- Desktop GUIs
 - - biblioteca Tk GUI
 - - bibliotecas separadas
 - - wxWidgets
 - - Kivy, para aplicações multitouch
 - - Qt, via pyqt ou pyside

Domínio da Aplicação

- Desenvolvimento de software
 - suporte ao desenvolvimento de software, para gerir e controlar a arquitetura, testes e muitas outras.
 - SCons para controlar arquitetura
 - Build e Apache Gump para compilação e teste contínuo e automatizado
 - Roundup ou Trac para rastrear bugs e desenvolver projetos.

Domínio da Aplicação

M

- SciPy .: matemática, ciência e engenharia.
- Panda .: análise de dados e modelagem.
- Ipython.: edição e registro de sessões de trabalho, programação paralela.

Tipos de Dados

- Tipos primitivos:
 - (int, long, float, complex) e cadeias de caracteres (strings)
 - listas, dicionários, tuplas e conjuntos
- Tipos definidos.

Variáveis

- São declaradas dinamicamente.

```
>>> mensagem = "E aí, Doutor?"  
>>> n = 17  
>>> pi = 3.14159
```

Listas, Tuplas, Strings e Dicionários

- Uma **lista**

```
>>> [10, 20, 30, 40]  
>>> ['spam', 'bungee', 'swallow']
```

- String

```
>>> fruta = "banana"  
>>> letra = fruta[1]  
>>> print letra
```

- Tuplas

```
>>> tupla = ('a', 'b', 'c', 'd', 'e')  
>>> tupla[0]  
'a'
```

- Dicionários

```
>>> ing2esp = {}  
>>> ing2esp['one'] = 'uno'  
>>> ing2esp['two'] = 'dos'
```

Classes

- Podemos definir um novo tipo composto, também chamado uma **classe**.

```
class Ponto:  
    pass
```

Objetos

- Representa uma instância de uma classe.

```
final = Ponto()
```

Funções Puras

- É bem parecido com os conceitos de funções em C

```
import math

def area(raio):
    temp = math.pi * raio**2
    return temp
```


Métodos

- Métodos são simplesmente como funções, com duas diferenças

```
class Horario:  
    pass  
  
    def exibeHora(time)  
        print str(time.horas) + ":" + \  
              str(time.minutos) + ":" + \  
              str(time.segundos)
```

Exceções

- São usadas para tratar de acontecimentos indesejáveis durante a execução do programa.

```
nomedoarquivo = raw_input('?Entre com o nome do arquivo: ?')
try:
    f = open (nomedoarquivo, '?r?')
except:
    print '?Não existe arquivo chamado?', nomedoarquivo
```

Legibilidade



- Bastante legível
- Indentação
- Poucos caracteres não alfanuméricos
- Aspecto didático
- Queda de legibilidade: tratamento de exceções
- Queda de interpretação: não identificar variáveis

Capacidade de Escrita

- Linguagem simples, poderosa e expressiva
- Alta capacidade de abstração
- Boa ortogonalidade
- Capacidade de escrita: alta

Confiabilidade

- Prejudicada pela verificação de dados
- Tratamento de exceções: orientada a objetos
- Modelo elegante de objetos
- Prejudicada pela alta flexibilidade -> contrabalanceia
- No geral, baixa confiabilidade (verificação de tipos)

Custo

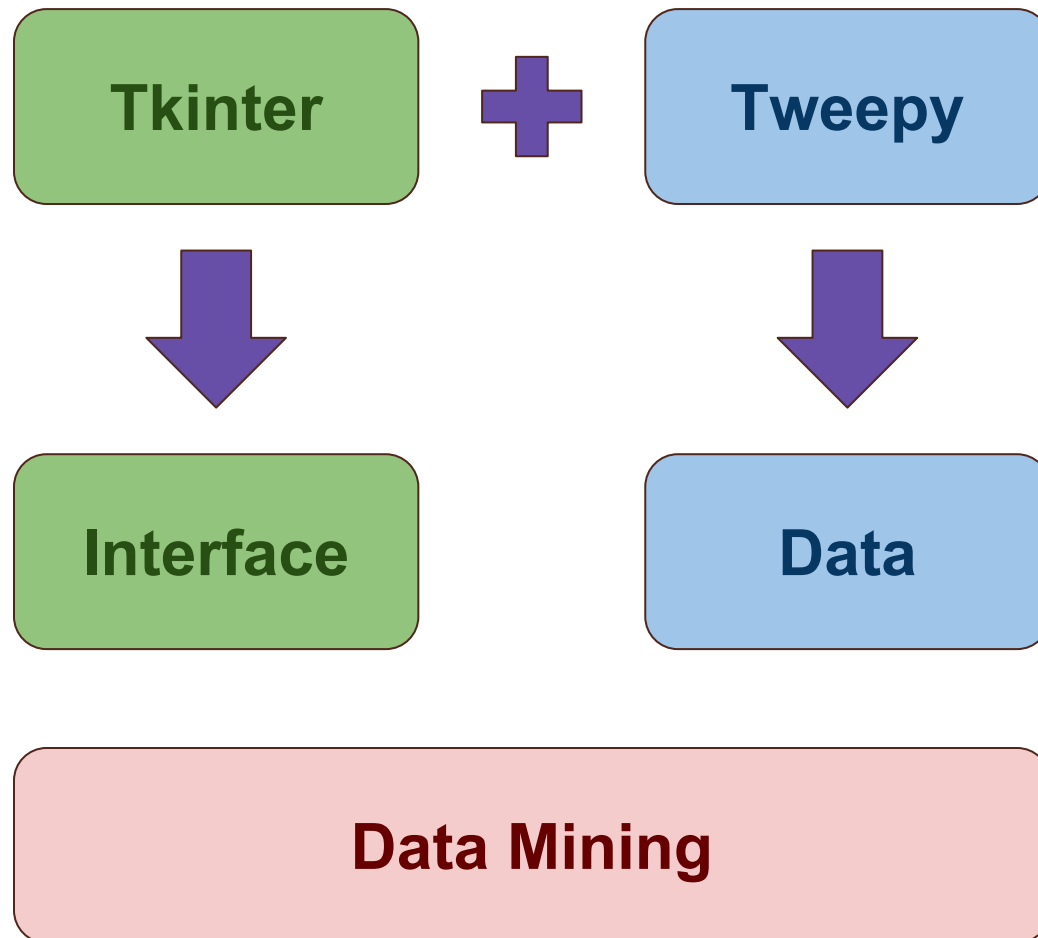
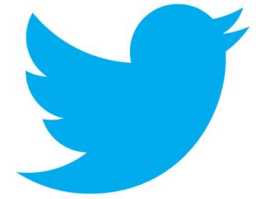
A

- Treinamento : baixo
- Escrever programas e manutenção: baixo
- Compilação e execução -> linguagem interpretada
- Documentação: bem definida, Open Source
 - <http://wiki.python.org.br/DocumentacaoPython>
- É uma linguagem de custo baixo

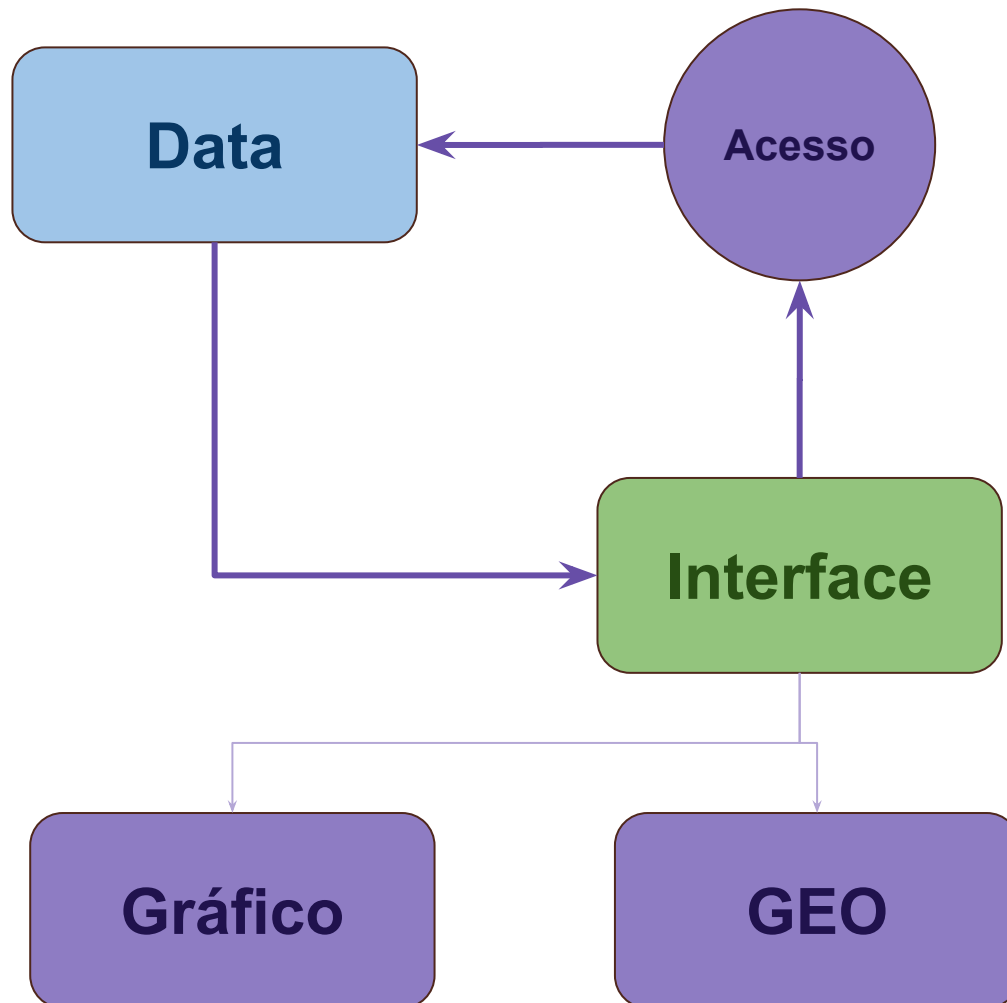
Portabilidade

- Executa em todas as principais plataformas em uso atualmente
- Interpretador e bibliotecas: implementação portátil
- Programas: código fonte portátil (versão Python instalada)
- No geral, a portabilidade tem crescido nos últimos anos

O Projeto



O Projeto



Justificativas

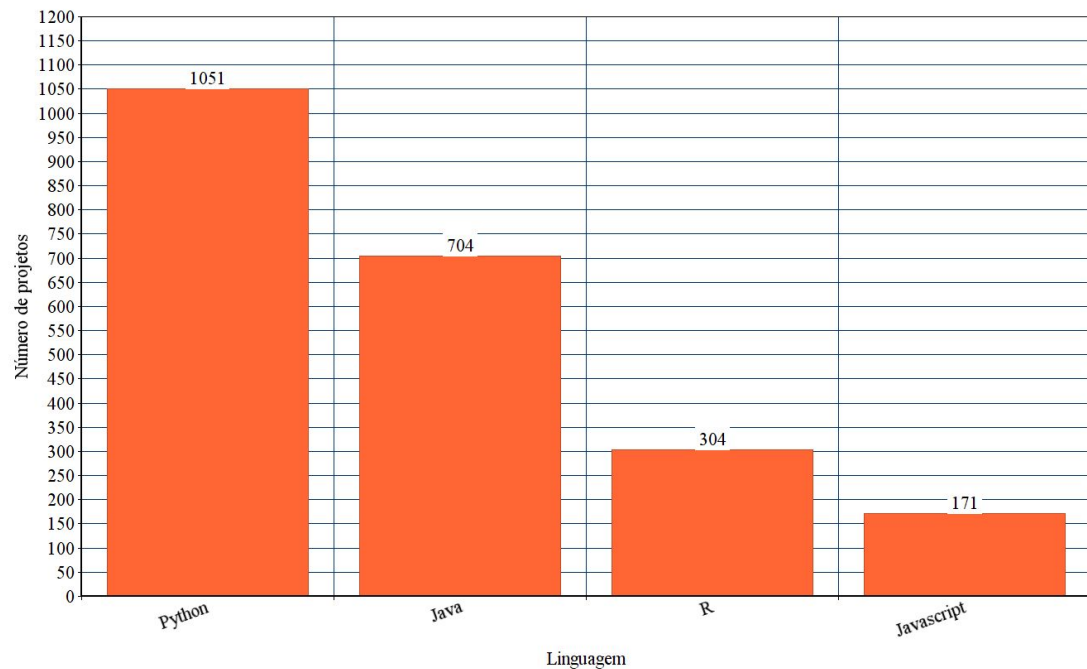
A

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
```

```
from tkinter import *
```

Pesquisa por "Data Mining" no GitHub

GitHub



Justificativas

```
terms_hashtag = [termo for termos in texto_twitter if termo.startswith('#') and termo not in palavras_proibidas]
```

```
for jsons in file_:
```

```
if e2.get() is not None and e2.get() is not "":
```

```
palavras_proibidas = palavras_retiradas+[palavra]
```

```
global palavras_retiradas
```

Especificidades

M

```
import threading  
  
import re  
  
import json  
  
from collections import Counter  
  
from tkinter import *  
  
from tweepy.streaming import StreamListener  
from tweepy import OAuthHandler  
from tweepy import Stream  
  
import vincent  
  
import webbrowser
```

Chart.html

Map.html

Paradigmas

A

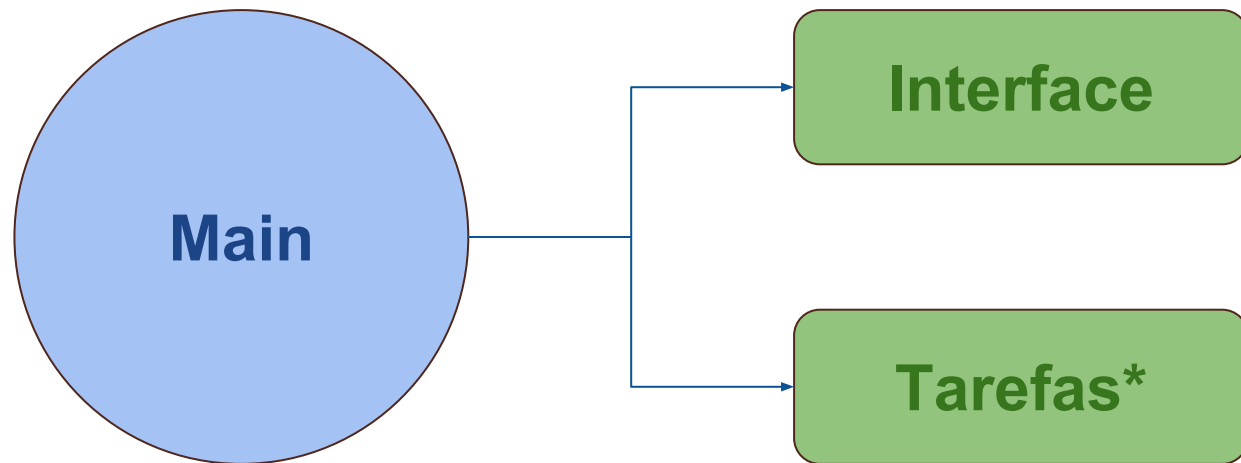
- Orientado à Objeto;

```
class uiThread (threading.Thread)  
class processThread (threading.Thread)  
class Leitor_twitter(StreamListener)
```

- Imperativo.

```
Programa = uiThread(2, "Programa", 2)  
Programa.start()
```

Uso de Threads



*Não conflitantes com interface!

Uso de Threads

F

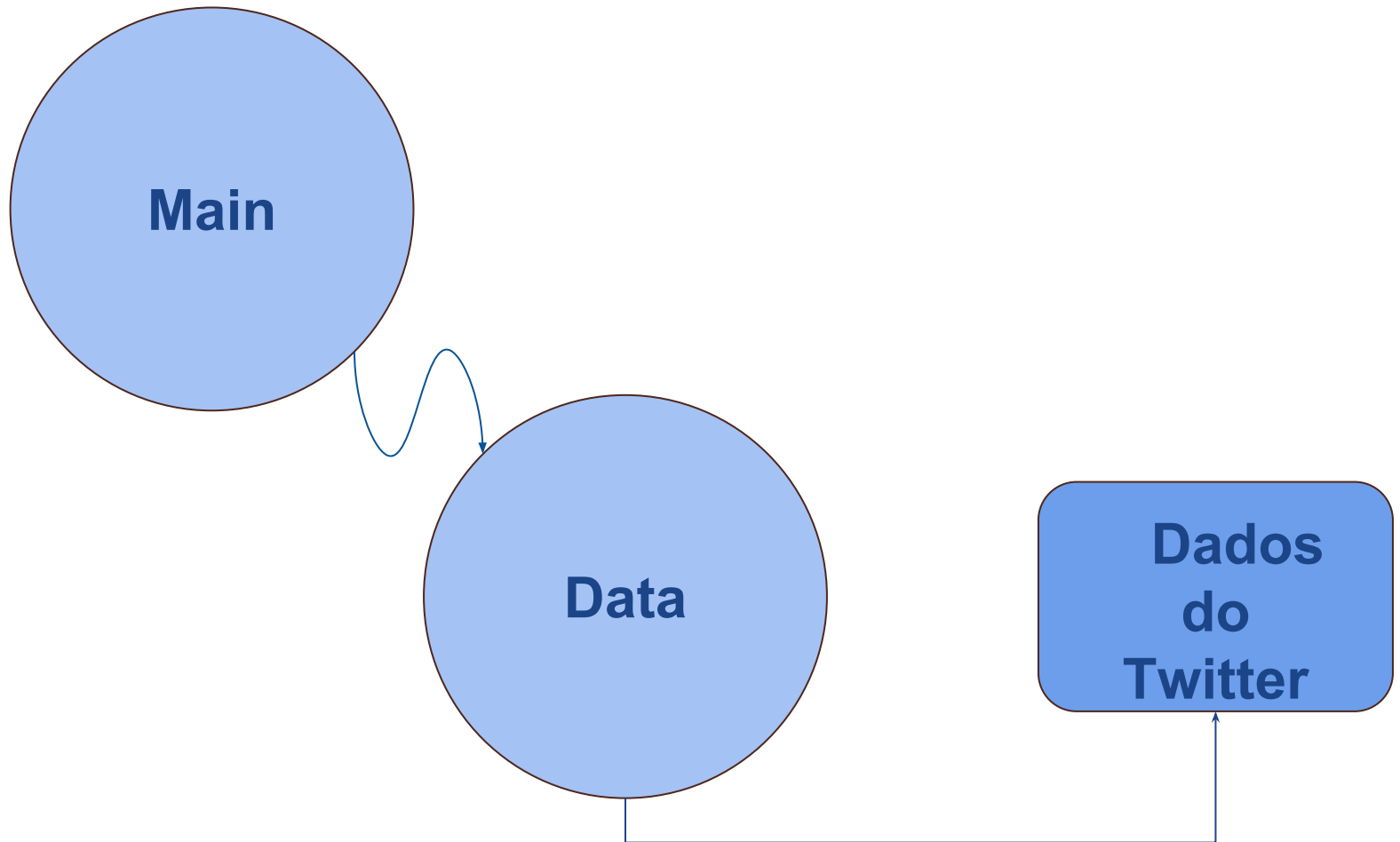
Interface

vs

**Dados
do
Twitter**

Uso de Threads

F





Implementação

Thread Principal

```
class uiThread (threading.Thread): #Implementa
threading.Thread
#
    def __init__(self, threadID, name, counter):

        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
```

Thread Principal

```
def run(self)
```

Interface

```
def entrada_acesso(self)
```

```
def getaccess_secret(self)
```

```
def getaccess_key(self)
```

```
def getcons_secret(self)
```

```
def getcons_key(self)
```

Acesso

```
def key(self)
```

```
def getkey(self)
```

Chave

Thread Principal

```
def data_acquire(self)  
def stop(self)
```

Dados



```
def data_open(self)  
def leitura_arquivo(self)
```

Abrir
Arquivo

Thread Principal

```
def token_type(self)
def tokens_hashtags(self)
def tokens_mentions(self)
def tokens_all(self)
```

**Tipos de
Tokens**

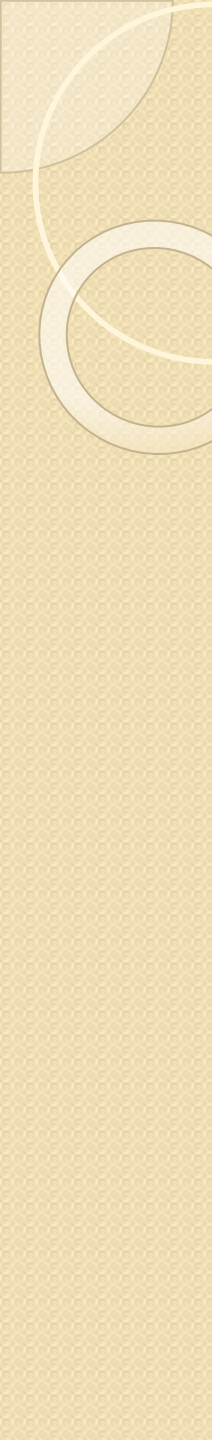
```
def token_num(self)
def getnumtk(self)
```

**Número de
Tokens**

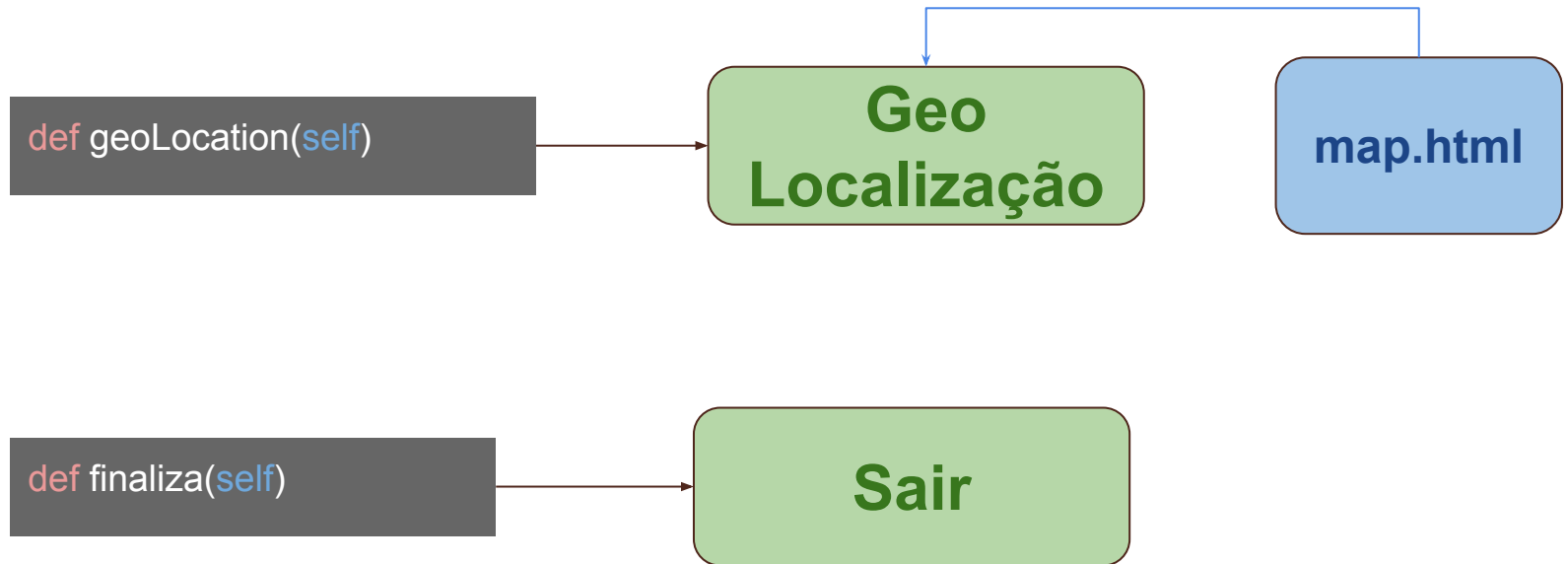
```
def tokenizer(self)
```

**Tokens
Frecuentes**

chart.html



Thread Principal



Thread Secundária

```
class processThread (threading.Thread):  
  
    def __init__(self,threadID, name, counter):  
        #  
        global file_  
        global arquivo_nome  
  
        threading.Thread.__init__(self)  
        self.threadID = threadID  
        self.name = name  
        self.counter = counter  
        file_ = open(arquivo_nome,'w')
```

Thread Secundária

```
def run(self)
```

Inicia Leitor

```
graph LR; A[def run(self)] --> B(Inicia Leitor)
```

The diagram illustrates the execution flow of a secondary thread. A dark gray rectangular box on the left contains the code snippet `def run(self)`. A horizontal arrow points from the right side of this box to a light green rounded rectangular box on the right. This green box contains the text **Inicia Leitor** in bold green font, indicating the action performed by the thread's run method.

Classe Leitor_twitter

```
def Leitor_Stream(self)
```

**Configura e
Inicia Leitor**

```
def on_data(self, dado)
```

**Recebe
Dados**

```
def on_error(self, status)
```

Erro

Perguntas?

F

A

M

G