PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Linguagem de Programação Python*

Seminário de Linguagem de Programação

Bruno Oliveira Santiago[1]
Fábio Freire Kochem[2]
Fernando Campos Silva Dal Maria[3]

Resumo

A popularidade do Python é evidenciada pelo fato de ser uma das linguagens mais utilizadas atualmente, juntamente com o JavaScript, outra linguagem amplamente conhecida na programação web. Embora a resposta para o motivo pelo qual Python se tornou tão popular seja complexa devido à diversidade de linguagens disponíveis, várias qualidades se destacam. Entre elas, destacam-se a qualidade do software produzido, a produtividade de desenvolvimento proporcionada pela linguagem, a portabilidade dos programas, o suporte a bibliotecas e a integração de componentes.

Python é valorizado por sua ênfase na legibilidade, coerência e tamanho de código reduzido. Além disso, sua capacidade de execução em diferentes plataformas de computadores e sua vasta coleção de funcionalidades portáteis contribuem para a atratividade da linguagem. Sua aplicação em programação web é particularmente notável, impulsionada por suas características favoráveis e pela ampla disponibilidade de recursos e bibliotecas para desenvolvimento web. Além dessa aplicação, ressaltamos também a presença da linguagem ná área técnico científica, com aplicações em inteligência artificial e análise de dados.

Este estudo visa fornecer uma compreensão abrangente das razões por trás do sucesso e popularidade da linguagem Python, destacando suas principais qualidades e vantagens para desenvolvedores web e além. Ademais fornecer uma compreensão detalhada sobre os paradigmas e construções sintáticas da linguagem. A análise apresentada neste artigo contribui para o entendimento do papel significativo desempenhado por Python no cenário atual da programação.

Palavras-chave: Python. Paradigmas. Aplicações. Popularidade.

---

*Artigo apresentado ao Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais como pré-requisito para obtenção do título de Bacharel em Ciência da Computação.

[1]Acadêmico do curso de Graduação em Ciência da Computação, Brasil– bruno.santi.oli@gmail.com.
[2]Acadêmico do curso de Graduação em Ciência da Computação, Brasil– fabiofkochem@gmail.com.
[3]Acadêmico do curso de Graduação em Ciência da Computação, Brasil– fernandocsdm@gmail.com.

1 INTRODUÇÃO

Python é uma linguagem de programação de propósito geral que é frequentemente aplicada em programação web. É comumente definida como uma linguagem de script orientada a objetos, uma definição que combina suporte a OOP com uma orientação geral em relação a funções de script. Sua estrutura é conhecida por combinar paradigmas procedural, funcional e orientado a objetos, uma afirmação que captura a riqueza e o escopo do Python atual. O fato de existir diversas linguagens atualmente dificulta a resposta do porque python se tornou tão popular, mas certamente existem diversas qualidades que chamam a atenção para a linguagem. Alguns deles são: qualidade de software, produtividade de desenvolvimento, portabilidade de programas, suporte à biblioteca e integração de componentes que enaltecem pontos fortes do Python como foco em legibilidade, coerência, tamanho de código reduzido, execução em diferentes plataformas de computadores e sua coleção extensa de funcionalidades portáteis (LUTZ, 2013). Veja na Figura 1 que Python é uma das linguagens mais populares atualmente, assim como Javascript, outra linguagem famosa na programação web.

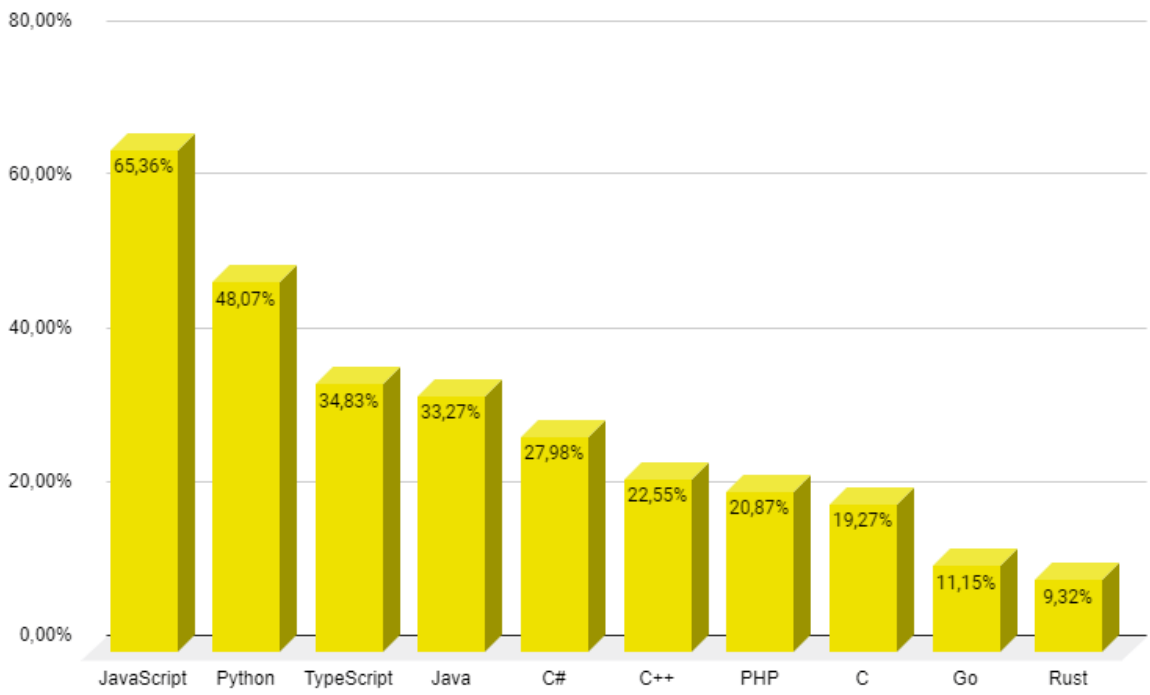

Figura 1 – Linguagens de programação mais populares 2022. Porcentagem de utilização na comunidade (STACKOVERFLOW, 2022).

1.1 O que é python?

Python é uma linguagem de programação interpretada multi-paradigma muito relevante para a ciência da computação. Essa é uma linguagem dinâmica criada com a intenção de au-

mentar o alcance da programação para a comunidade Unix/C. Python foi criada com a intenção de melhorar aspéctos característicos da linguagem ABC, antes desenvolvida pela equipe da Centrum Wiskunde & Informatica, para facilitar a confecção de utilitários para o sistema operacional Amoeba. Guido pretendia aprimorar alguns aspectos da linguagem ABC como I/O (entrada e saída), características restritivas: criação e uso de tipos, sintaxe ideossincrática (todas as palavras-chave em letras maiúsculas) e seu editor estruturado integrado, que seus usuários reprovavam. Python acabou se tornando uma das linguagens mais utilizadas atualmente e é amplamente reconhecida por ser uma lingagem com alta portabilidade, alto poder computacional, integrável, dinâmica e de fácil aprendizado [(ROSSUM, 1996), (DAYLEY, 2006)].

1.2 Aplicações da linguagem

Python é utilizada tanto na área acadêmica quanto na área empresarial. Ela esta presente em diversas aplicações, dentre elas: aplicações web, aplicações científicas, matemáticas (por exemplo: análise de dados e inteligência artificial), educacionais e desenvolvimento de interfaces gráficas. Sua abrangência nas mais diversas áreas pode ser justificada por diversos fatores, alguns dos quais já foram listados na parte inicial deste capítulo. Entretanto, não poderiamos deixar de citar outros caracteísticas como seu suporte à aritmética avançada e à diversos tipos de arquivos, além do suporte a diversas bibliotecas e frameworks [(ROSSUM, 1996), (DAYLEY, 2006), (PYTHON SOFTWARE FUNDATION, 2023a)]. Observe na Tabela 1 algumas das mais conhecidas e utilizadas bibliotecas de python, assim como diversos frameworks web.

Framework	Aplicação	Biblioteca	Aplicação
Bottle	Web Apps	NumPy	Algebra
Django	Web Apps	OpenCV	Computer Vision
Flask	Web Apps	Pandas	Graphics
Kivy	Apps Development	PyGame	Games
Pylons	Web Apps	Requests	Web Apps
Tensorflow	Machine Learning	PySide	App Development

Tabela 1 – Alguns do frameworks e bibliotecas de python (PYTHON SOFTWARE FUNDATION, 2023a).

Como exemplificado na Tabela 1, Python apresenta diversos frameworks e bibliotecas, dentre os quais, aqueles voltados a programação de aplicações web saltam aos olhos. Veja, por exemplo, o micro-framework Flask, conhecido por proporcionar simplicidade, flexibilidade e escalonabilidade a projetos de servidores web. Esse freamework foi e ainda é bastante utilizado para a criação de projetos e fornece ao desenvolvedor uma gama de possibilidades e funcionalidades para integração e aumento da eficiência da aplicação. Flask depende do kit de ferramentas Werkzeug WSGI, do mecanismo de modelo Jinja e do kit de ferramentas Click CLI. Note que nesse famework é possível confeccionar uma aplicação mínima em apenas 5 linhas de código, como na Figura 2 [(FLASK, 2023), (VOGEL et al., 2017)].

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Figura 2 – Aplicação mínima em Flask (FLASK, 2023).

Outro exemplo é o framework Django, utilizado para criar complexos sistemas web em Python. Esse framework é conhecido por ser rápido, escalável, seguro, versátil e abrangente, contendo ferramentas para cuidar da autenticação do usuário, administração de conteúdo, mapas do site, feeds RSS e muitas outras tarefas (DJANGO, 2023). Django, assim como falsk é um dos principais frameworks web para Python e é muito importante na programação web. Um exemplo de uso deste framework é retratado na pesquisa de Ishan Adhikari Bairagi (et al.) que apresenta o processo de desenvolvimento e a análise de um projeto estruturado com HTML, Django e Bootstrap (BAIRAGI et al., 2021). Ressaltamos aqui que informações sobre diversos outros frameworks e bibliotecas para desenvolvimento web com Python podem ser encontradas na Wiki oficial do Python (PYTHON WIKI, 2023).

Observe, entretanto, que com o avanço da internet, diversos sites estão encontrando problemas envolvendo multiplas solicitações de usuário e a necessidade de alta simultaneidade e tratamento de concorrência para as requisições. Nesse cenário, a linguagem dinâmica Javascript cresceu em popularidade e em utilização, veja, por exemplo, na Figura 3 que tecnologias como Node.js e frameworks como ReactJS e JQuery (feitos para Javascript) são consideravelmente mais populares e mais utilizados do que aqueles voltados para Python (Django e Flask), já retratados nos parágrafos anteriores. Lembre-se que, como dito anteriormente, o processamento eficiente de múltiplas requisições concorrentes é de extrema importância na web atual, logo avaliar a situação através de uma visão voltada para a performance das diferentes linguagens pode proporcionar uma justificativa para a maior popularidade das tecnologias Javascript [(CHALLAPALLI et al., 2021), (STACKOVERFLOW, 2022)].

Para analisar melhor a situação recorremos a pesquisa realizada por Sai Sri Nandan Challapalli (et al.), que foca em analisar o processo de desenvolvimento de sites estáticos e dinâmicos que utilizam Python e Node.js para o back-end. O artigo realiza uma comparação de desempenho entre as duas tecnologias a partir de duas métricas: (1) respostas à requisições por segundo e (2) tempo de resposta para cada requisição. Os resultados obtidos mostraram que o servidor Node.js performou significativamente mehor do que o servidor em Python. O servidor em Node.js obteve uma performance, aproximadamente, 250 vezes maior do que aquela obteida pelo servidor Python. Durante os testes Node.js atendeu, em média, a 30616 requisições em um período de 30 segundos enquanto o servidor Python atendeu a apenas 121. Observe também, na Figura 4 o tempo médio por resposta a requisição obtidos em cada servidor (CHALLAPALLI et al., 2021).

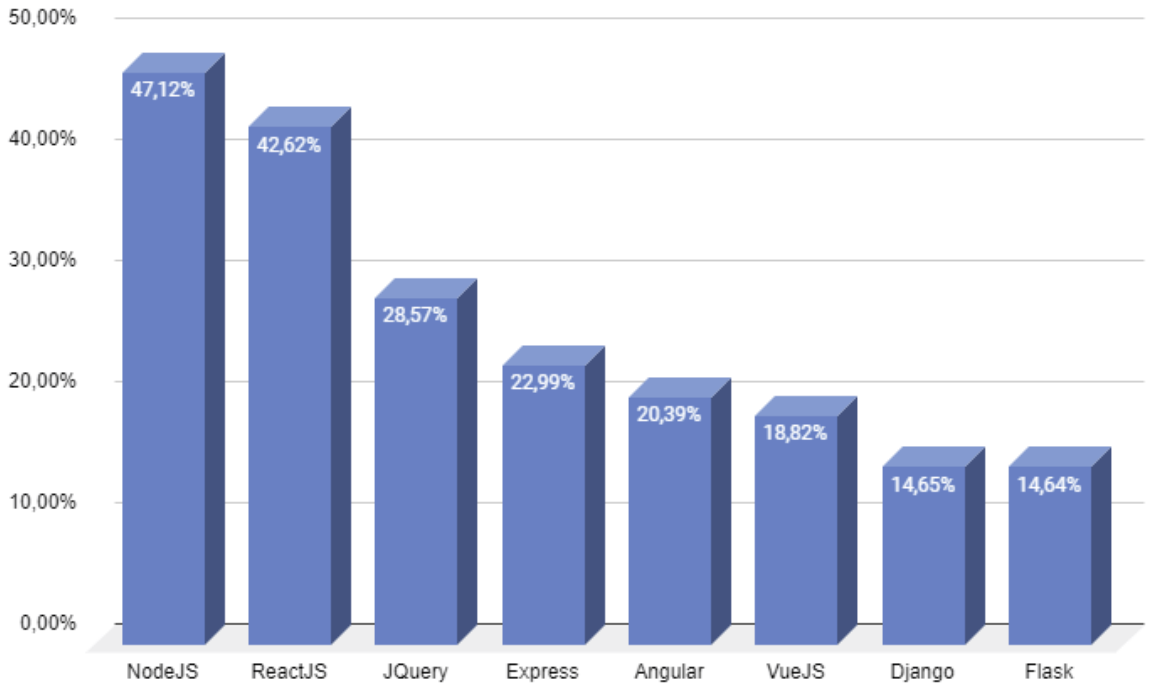

Figura 3 – Frameworks web mais popoulares 2022. Porcentagem de utilização na comunidade (STACKOVERFLOW, 2022).

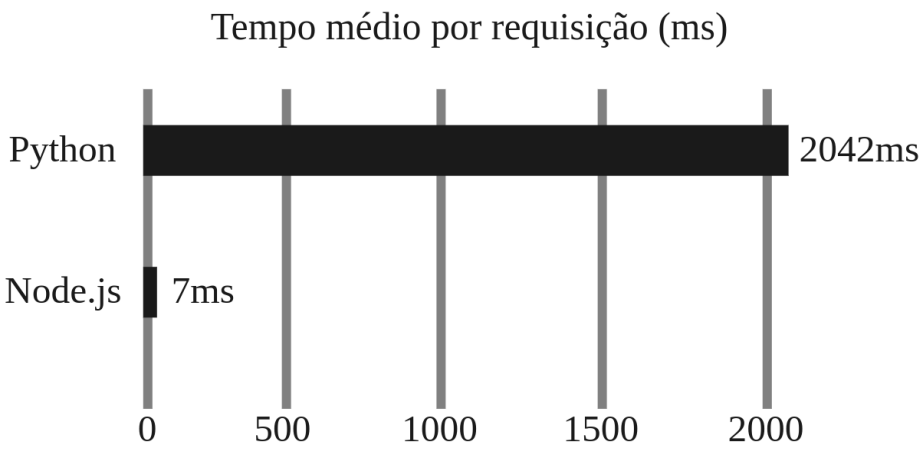

Figura 4 – Tempo médio por resposta a requisição em milisegundos(ms) (CHALLAPALLI et al., 2021) - Adaptado.

Por fim ressaltamos que, mesmo que Python não seja a principal escolha para programação web, atualmente a linguagem está sendo muito utilizada para análise de dados, programação científica e tecnica, além de também ser muito aplicada na área de Inteligência Artificial. As linguagens METALAB e R, por exemplo, são muito utilizadas para aplicações científicas e Python se tornou uma famosa alternativa para essas linguagens. Essa preferência pela lingaugem se deve a sintaxe simples, modularizada e com design voltado para programação orientada a objetos, ademais a presença de bibliotecas como Numpy (já citada na Tabela 1) enfatiza a linguagem para aplicações númericas (NAGPAL; GABRANI, 2019). Veja na Figura 5 que Numpy, Pandas e Tensorflow, forma as bibliotecas mais trabalhadas e populares no ranking

do StackOverflow (STACKOVERFLOW, 2022). Ressaltamos também a presença de Python em várias faculdades, incluindo a Stanford, UFMG e PUC-Minas, como recurso para lecionar matérias que requerem aplicação matemática, como Inteligencia Artifivial.

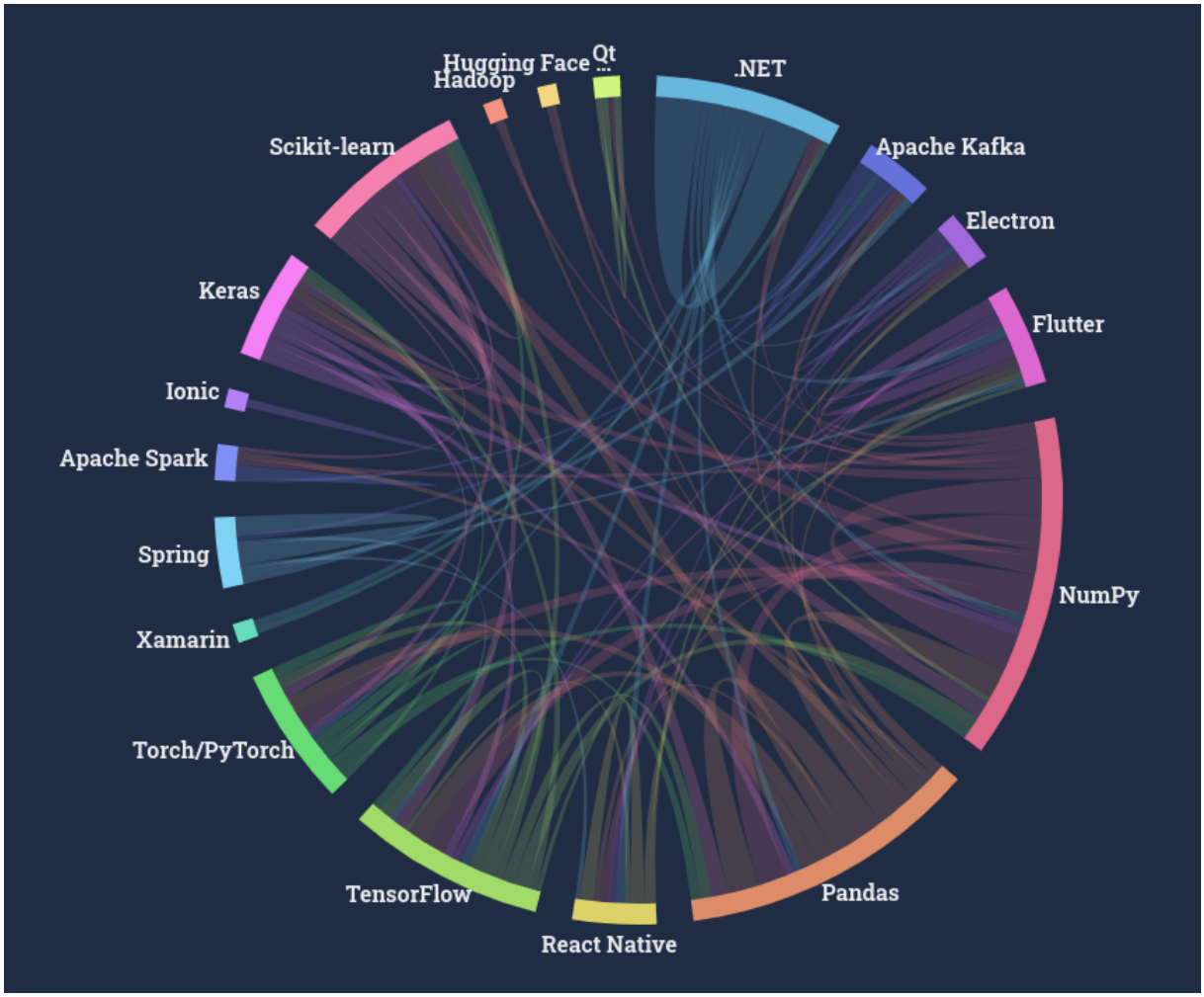

Figura 5 – Bibliotecas mais populares e utilizadas pela comunidade 2022 (STACKOVERFLOW, 2022).

2 HISTÓRICO DA LINGUAGEM

A linha do tempo das linguagens de programação é algo muito extenso, uma vez que as primeiras linguagens de programação a existirem no mundo foram criadas há mais de 50 anos, resultando em uma longa história com diversos tipos de implementação e propósitos. No entanto, algumas linguagens marcaram a genealogia das linguagens de programação de alto nível comuns, sendo algumas delas: Fortran, Pascal, ML, Haskell, C, C++, Python, Java, C# e JavaScript [(LUTZ, 2013), (SEBESTA, 2019)].

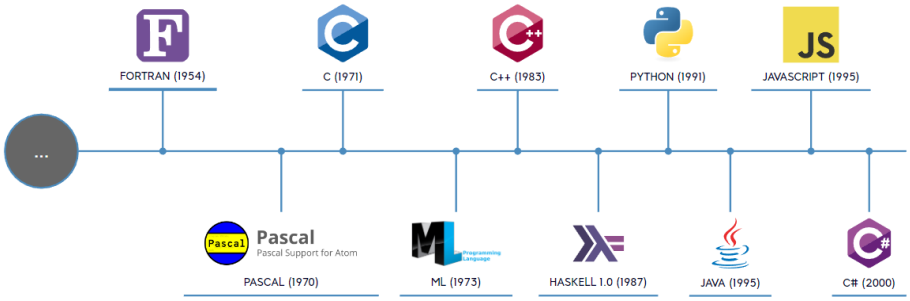

Figura 6 – Linha do Tempo das Linguagens de Programação

2.1 O Criador

Guido van Rossum é um renomado programador holandês, famoso por criar a linguagem de programação Python. O distinto engenheiro, que atualmente trabalha na divisão de desenvolvimento da Microsoft, teve uma longa e brilhante carreira na computação [(ROSSUM, 1996), (SEVERANCE, 2015a), (ROSSUM, 2023)]. É o criador da segunda linguagem mais famosa entre os desenvolvedores, a linguagem Python. Uma linguagem baseada principalmente nas linguagens ABC, C e Modula-3 [(STACKOVERFLOW, 2022), (SEBESTA, 2019)].

2.2 A linguagem ABC (80s)

No final dos anos 80 Guido van Rossum trabalhou no Centrum Wiskunde & Informatica (CWI) desenvolvendo um sistema operacional distribuido chamado Amoeba. Como o modelo do sistema de arquivos do Amoeba era diferente do modelo exitente Unix, novos utilitários teriam que ser cofeccionados pela pequena equipe responsável pelo projeto. Confeccionar tantos utilitários levaria um tempo considerável utilizando a linguagem C, portanto Guido se perguntou se não seria possível criar uma nova linguagem que facilitaria a confecção dos utilitários [(SEVERANCE, 2015a), (ROSSUM, 1996)].

Com esse intuito Guido trabalhou na confecção da linguagem ABC, uma linguagem

muito promissora para a época, entretanto seu alto nível de abstração tornava a linguagem inadequada para conversar com outros sistemas. Durante uma longa pausa de férias em dezembro de 1989, Guido começou a desenvolver uma linguagem semelhante a ABC que poderia se comunicar com o sistema operacional e seria adequada para o desenvolvimento rápido de utilitários de sistema operacional para o Amoeba. Ele chamou seu projeto nascente de Python, inspirando-se no programa de televisão Monty Python's Flying Circus [(SEVERANCE, 2015a), (ROSSUM, 1996)].

Figura 7 – Monty Python's Flying Circus Logo

2.3 A Primeira Versão (1991)

A primeira versão do Python foi lançada em fevereiro de 1991 e distribuída usando o grupo de notícias alt.sources. Com o aumento de engajamento da comunidade algumas empresas começaram a utilizar e até mesmo depender de Python. Nesse momento o instituto NIST (National Institute of Standards and Technology) convidou Guido para para os Estados Unidos, local onde o primeiro workshop de Python foi sediado em novembro de 1994. Por meio do workshop Python, Guido conheceu a CNRI (Corporation for National Research Initiatives) empresa para a qual trabalharia durante os anos de 1995 a 2000 [(SEVERANCE, 2015a), (ROSSUM, 1996)].

A CNRI possibilitou que Guido e sua equipe fizessem levassem Python adiante, com investimentos significativos, mas alguns se perguntaram se o produto deveria mudar de código aberto para um modelo em que algum lucro pudesse ser obtido. Situação que levou Guido a entrar em conflito com a liderança do CNRI sobre o licenciamento da linguagem [(SEVERANCE, 2015a), (SEVERANCE, 2015b), (ROSSUM, 1996)].

2.4 A Segunda Versão (2000)

Enquanto um conflito de interesses estava acontecendo na CNRI uma pequena startup de código aberto entrou em contato com Guido e lhe ofereceu uma vaga para trabalhar com

Figura 8 – Python Logo (PYTHON SOFTWARE FUNDATION, 2023c)

Python e com uma pequena equipe em tempo integral. Enquanto trabalhava na startup Guido e sua equipe criaram o Python 2 (lançado em outubro de 2000) que era muito importante, tato por utilizar Unicode quanto por representar uma quebra de vínculo definitiva com a empresa CNRI, que estava em desacordo quanto à propriedade de parte do código da linguagem [(SEVERANCE, 2015a), (SEVERANCE, 2015b), (ROSSUM, 1996)].

A medida que os conflitos de licenciamento com CNRI eram resolvidos, a startup na qual Guido trabalhava começou a desmontar. Após o fim da startup Guido foi trabalhar para uma empresa de sucesso fortemente envolvida com a comunidade Python, a empresa Zope. Como esta empresa era focada na linguagem ela se benficiava com seu investimento em Guido e em sua equipe [(SEVERANCE, 2015b), (ROSSUM, 1996)].

2.5 Python 3000 (2008)

Por volta de 2005, com o aumento das aplicações Python e com o aumento dos problemas de compatibilidade, a linguagem precisou de uma revisão. Os problemas encontrados na linguagem poderiam ser concertados e a equipe que realizaria o projeto ja apresentava as soluções ealboradas, entretanto o concerto tornaria a linguagem incompatível com suas versões anteriores. Nesse momento surgio a ideia de confeccionar o Python 3000 (ou "py3k") que se tornaria o atualmente conhecido Python 3. A linguagem lançou sua terceira versão em dezembro de 2008 [(SEVERANCE, 2015b), (ROSSUM, 1996)].

3 PARADIGMAS DE PYTHON

Uma linguagem de programação (LP) é uma linguagem formal voltada para a descrição e aplicação de algoritmos. As LPs são voltadas para a resolução de problemas computacionais, que por sua vez, podem ser solucionados dependendo da perspectiva do desenvolvedor/pesquisador. Por essa razão diferentes paradigmas de linguagem foram elaborados e são selecionados de acordo com o problema computacional a ser trabalhado. Os paradigmas de programação declarativo Funcional (FP) e imperativo Orientado a Objetos (OOP) são os mais comuns [(WANG; ZHANG, 2022), (SEBESTA, 2019)]. Veja na Tabela 2 algumas das linguagens de programação mais utilzadas, assim como seu paradigma correspondente e seus cenários de aplicação.

Tabela 2 – Diversos tipos de linguagens

(WANG; ZHANG, 2022)

Language	Programming Paradigm	Type System	Compilation Mode	Memory Model	Release Date	Application Scenarios
Python	Multi-paradigm	Dynamically, Strongly	JIT	GC	1991	Web, Enterprise, Embedded
Java	Multi-paradigm	Statically, Strongly	AOT	GC	1995	Web, Mobile, Enterprise
C++	Multi-paradigm	Statically, Weakly	AOT	Manual	1983	Mobile, Enterprise, Embedded
JavaScript	Multi-paradigm	Untyped	JIT	GC	1995	Web
Go	Multi-paradigm	Statically, Strongly	AOT	GC	2009	Web, Enterprise
Swift	Multi-paradigm	Statically, Strongly	AOT	ARC	2014	Mobile, Enterprise
Dart	Multi-paradigm	Statically, Strongly	AOT&JIT	GC	2011	Web, Mobile
Rust	Multi-paradigm	Statically, Strongly	AOT	Ownership	2015	Web, Enterprise, Embedded
Kotlin	Multi-paradigm	Statically, Strongly	AOT&JIT	GC	2016	Web, Mobile

Como visto na Tabela 2, múltiplas LPs modernas são multi-paradigma e consequentemente apresentam diferentes graus de suporte a conceitos de paradigmas diferentes. Por exemplo, algumas LPs lançadas por volta de 2010 já apresentam suporte aos conceitos centrais do PF, algumas dessas linguagens são Rust e Kotlin (WANG; ZHANG, 2022).

Tabela 3 – Graus de suporte a conceitos do paradigma funcional

(WANG; ZHANG, 2022)

Language	Higher-order functions	Lambda expression	Partial application	Closures	Type inference	Pattern matching	Statements as expressions
Python	✓	✓	Python2.5	✓	✓	Python3.10	×
Java	Java8	Java8	Java8	Java8	Java10	×	×
C++	C++11	C++11	C++11	C++11	C++11	C++17	×
JavaScript	✓	✓	ECMAScript5	✓	✓	ECMAScript6	×
Go	✓	✓	✓	✓	✓	×	×
Swift	✓	✓	✓	✓	✓	✓	×
Dart	✓	✓	✓	✓	✓	×	×
Rust	✓	✓	✓	✓	✓	✓	✓
Kotlin	✓	✓	✓	✓	✓	✓	✓

Language	Inheritance	Delegation	Traits	Everything is object
Python	✓	✓	×	✓
Java	✓	×	✓	×
C++	✓	×	✓	×
JavaScript	✓	✓	×	✓
Go	×	×	✓	×
Swift	✓	×	✓	✓
Dart	✓	×	×	✓
Rust	×	✓	✓	✓
Kotlin	✓	✓	✓	✓

Figura 9 – Graus de suporte a conceitos do paradigma de orientação a objetos (WANG; ZHANG, 2022).

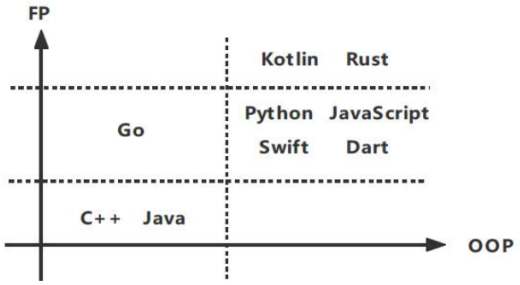

Figura 10 – Gráfico para o suporte a FP e a OOP (WANG; ZHANG, 2022).

3.1 Paradigmas de Python

Python é uma linguagem de programação multi-paradigma com suporte completo a orientação a objetos, fator explícito pelo fato de que nela todo valor representado é na verdade um objeto que contém diversos métodos pré-estabelecidos. Dentre os paradigmas suportados pela linguagem estão: (1) a programação funcional (FP), (2) programação orientada a objetos (OOP), (3) programação imperativa (PI), (4) programação procedural (PP) e (5) *aspect-oriented programming* (misto) (DYER; CHAUHAN, 2022).

Assim como outras linguagens multi-paradigma, Python contém um paradigma predominante, ao qual da suporte e enfatiza, entretanto apresenta também diversos recursos advindos de outros paradigmas, observer com mais detalhe esses recursos na Tabela 4. O paradigma predominante de Python é a orientação por objetos, portanto esse é o enfoque da linguagem (DYER; CHAUHAN, 2022). Para exemplificar a afirmação anterior observe que a linguagem apresenta uma método chamado "type()", que retorna o tipo de determinado valor. Caso o interpretador receba o método "type('Hello World')"a resposta será «class 'str'>"(PYTHON SOFTWARE FUNDATION, 2023b).

3.2 Paradigma Imperativo

Para o paradigma imperativo, o código deve descrever explicitamente o processo de execução de determinado algorítmo através de cláusulas que alteram o estado do programa. Esse paradigma é intrínseco da linguagem Python, principalmente porque é a base para OOP e PP. Vale ressaltar que ele também é presente em outras linguagens que influenciaram python, como em ABC e C (DYER; CHAUHAN, 2022). Para exemplificar, observe o trecho de código: `"num: int = 10"`, em Python essa construção é um exemplo de aplicação do paradigma imperativo (PYTHON SOFTWARE FUNDATION, 2023b).

3.3 Paradigma Procedural

A programação procedural consiste em organizar funcionalidades de um programa em trechos de código reutilizáveis chamados procedimentos. Esse paradigma proporciona as linguagens que o utilizam um maior grau de abstração, permitindo que os desenvolvedores diferenciem a forma como algo é implementado, de sua funcionalidade, funcionalidade de extrema importancia para uma linguagem de programação [(DYER; CHAUHAN, 2022), (SEBESTA, 2019)]. Em python, um procedimento é chamado de função e pode ser definido como se segue:

```
def func():
    num: int = 10
```

Figura 11 – Funcão em python (PYTHON SOFTWARE FUNDATION, 2023b).

3.4 Paradigma Orientação a Objetos

A programação orientada a objetos é um estilo de programação que fornece abstrações utilizando classes e objetos. Uma classe é uma estrutura que define o estado (campos) e o comportamento (métodos) de um objeto. Os objetos são instâncias de classes e podem ser criados e manipulados por referência. Ao utilizar a programação orientada a objetos em Python, qualquer declaração de classe, incluindo todo o seu conteúdo, é considerada uma aplicação desse paradigma. Também consideramos as chamadas de métodos em objetos como parte da programação orientada a objetos. Ao classificar o paradigma utilizado em um programa, é necessário analisar cada declaração dentro do corpo da classe. Se a declaração utilizar recursos funcionais ou recursos procedurais, isso indica a utilização de múltiplos paradigmas, e o programa é classificado como tal (DYER; CHAUHAN, 2022).

3.5 Paradigma Funcional

Programação funcional é um estilo de programação que se baseia na declaração do que se deseja que um programa faça, em vez de como ele deve ser executado, diferentemente dos estilos imperativos anteriores. Na programação funcional, as funções são a unidade fundamental. Em Python, as funções funcionais são chamadas de lambdas. Nesse paradigma, os usuários criam e combinam funções para construir seus programas. A iteração é um princípio essencial na programação funcional e geralmente é realizada por meio de funções recursivas. É relevante compreender a frequência com que os recursos funcionais são utilizados em Python, pois isso auxilia no direcionamento de futuras mudanças na linguagem. Vale ressaltar que existem várias Propostas de Aprimoramento do Python (PEPs) que foram aceitas ou estão sendo considera-

das para adicionar recursos funcionais adicionais, como a expressão geradora já implementada (PEP 289), o suporte a co-rotinas para geradores (PEP 342) e a proposta de correspondência estrutural (PEP 634) (DYER; CHAUHAN, 2022).

3.6 Classificação das Funcionalidades

Como visto anteriormente, Python apresenta diversos recursos advindos de diferentes paradigmas de programação, portanto apresentamos na Tabela 4 retirada de uma pesquisa realizada por Robert Dyer e Jigyasa Chauhan em 2022 acerca dos paradigmas presentes na linguagem.

Tabela 4 – Classificação dos recursos de Python de acordo com o paradigma a qual pertencem (WANG; ZHANG, 2022).

Python Language Feature	Imperative	Procedural	Object-Oriented	Functional[1]
`if else elif`	x			
`while` loop	x			
`break`	x			
`continue`	x			
`assert`	x			
`del`	x			
array indexing	x			
`pass` (inside loop)	x			
`pass` (inside class)	x		x	
`pass` (inside def)	x	x		
`return`		x		
function (`def`)		x		
nested function (`def`)		x		
`class` declaration			x	
inheritance			x	
method (`def`)			x	
`with`	x		x	
`try`	x		x	
`except`	x		x	
`finally`	x		x	
`raise`	x		x	
`for` loop	x			x
(`not`) `in` operator	x			x
`yield`	x			x
function-as-arg				x
`lambda` functions				x
list comprehension				x
decorators				x
generator expressions				x
iterators (`__next/iter__()`)	x	x	x	x
`send()` (into generator)		x		x
`iter()`		x		x
`map()`		x		x
`sorted()`		x		x
`filter()`		x		x
`any()`		x		x
`all()`		x		x
`itertools.*()`		x		x
`functools.*()`		x		x
`enumerate()`		x		x
`zip()`		x		x

4 ESTRUTURA DA LINGUAGE

Como já mencionado nos capítulos anteriores, Python é uma linguagem muito abrangente e apresenta muitas características marcantes que a distingue das demais linguagens utilizadas atualmente. Nesta seção detalhamos um pouco da estrutura da linguagem, suas construções sintáticas, alguns de seus mecanismos de abstração e algumas de suas características mais fortes. Desse modo, para iniciar esse capítulo apresentamos uma lista com as principais características da linguagem, algumas das quais já contemplamos neste trabalho.

- Multi-paradigma
- Estrutura de Bloco Aninhada
- Dinâmica e Fortemente Tipada - *Duck Type*
- Amarrada Dinâmicamente
- *Garbage Collected* - Utiliza um coletor de lixo
- **Simples e Legível**

4.1 Equivalência de tipos e *Duck Type*

Inicialmente, Python foi desenvolvida para apresentar equivalêcia de tipos por nomes, entretanto, com o avanço da linguagem, uma equivalência de tipo estrutural foi adicionada e ficou conhecida como *Duck Typing*. *Duck Typing* foi batizado em homenagem ao "teste do pato", desenvolvido por James Whitcomb Riley: *"If it walks like a duck, and it quacks like a duck, then it must be a duck"* ("Se um pássaro anda como um pato e grasna como um pato, então ele deve ser um pato"). Essa afirmação mostra que ao avaliar-se as funcionalidades e/ou os comportamentos de determinada entidade é possível inferir qual sua origem comportamental ou funcional. Ou seja, podemos realizar uma inferência de tipo baseada em determinado comportamento de um objeto (MILOJKOVIC et al., 2017).

Em Python essa análise é realizada através de métodos, que embora possam pertencer a classes distintas, apresentam a mesma assinatura. Note, portanto, que esse conceito também está relacionado com o polimorfismo apresentado pela linguagem, já que, mesmo que um objeto pertencente a uma classe A, não compartilhe a mesma superclasse que outro objeto da classe B ele pode ser utilizado em construções específicas da classe B, desde que apresente os métodos necessários para atendê-la (MILOJKOVIC et al., 2017). Python *duck type* permiti que o programador não seja limitado pela equivalência de tipos nominal, na qual ele seria obrigado a declarar explicitamente o tipo de determinada estrutura. Observe o exemplo da Figura 12.

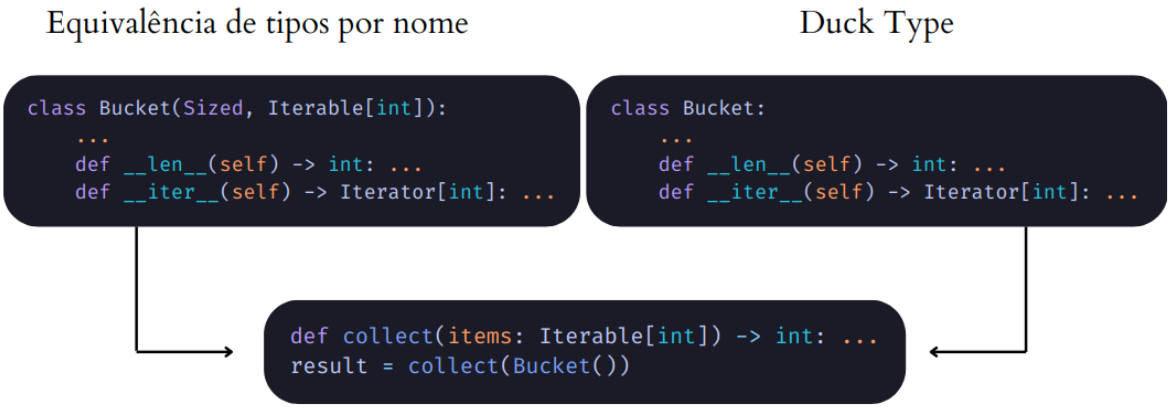

Figura 12 – Comparação entre um código com declarações explícitas de nomes de tipos e outro utilizando duck typing (confeccionado pelos autores deste artigo).

É importante ressaltar que *Duck Typing* é uma das principais características de Python que permitem que ela consiga implementar um sistema de tipagem dinâmico com uma forte verificação de tipos.

4.2 Tipos e Operadores

Python suportda diversos tipos primitivos e compostos, dentre eles podemos ressaltar um que não é suportado diretamente pelo hardware, o Long Integer que permite que Python realize opreações com números inteiros extremamente grandes. A linguagem também se diferencia por apresentar uma ampla gama de operadores que podem ser sobrecarregados pelo programador através do uso dos métodos especiais da linguagem (métodos __<nome_do_methodo>__) (PYTHON SOFTWARE FUNDATION, 2023b). Observe nas Figuras 13, 14 e 15 os tipos, os operadores aritméticos, bit a bit e lógicos, apresentados pela linguagem.

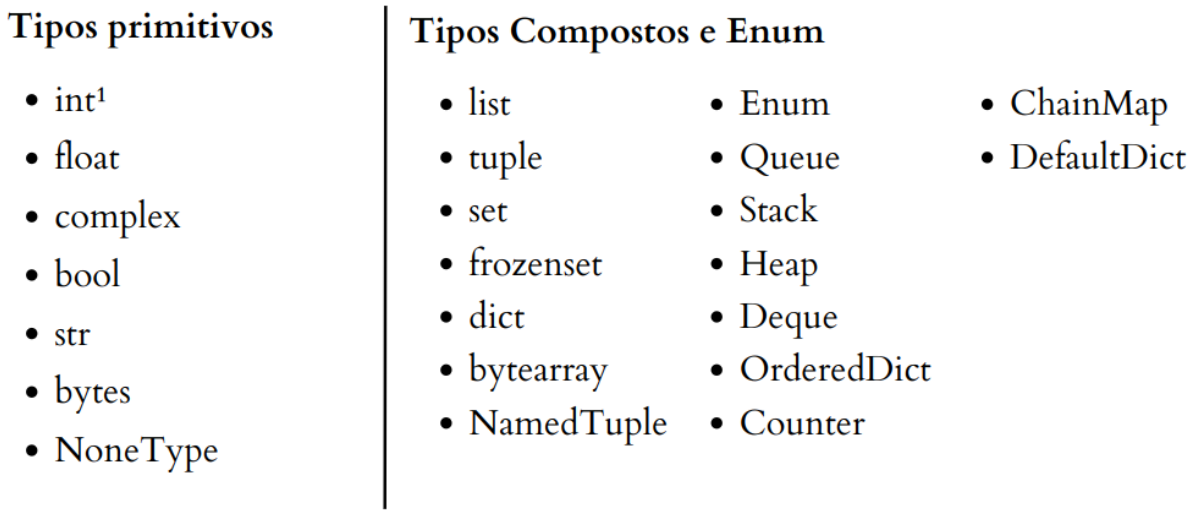

Figura 13 – Principais tipos de Python (PYTHON SOFTWARE FUNDATION, 2023b)

```
+ # adição
- # subtração
* # multiplicação
// # divisão de inteiros
/ # divisão de pontos flutuantes
** # potenciação
@ # multiplicação de matrizes
% # resto da divisão inteira
= # atribuição
:= # atribuí valor à um identificador quando em uma expressão condição
```

Figura 14 – Operadores aritméticos de Python (PYTHON SOFTWARE FUNDATION, 2023b)

```
# Bit a Bit
                    &  ^  |  ~  <<  >>
```

```
# Lógicos
            and  or  not  <=  <  >=  >  ==  !=
```

Figura 15 – Operadores bit a bit e lógicos de Python (PYTHON SOFTWARE FUNDATION, 2023b)

4.3 Controle de Fluxo, Iteráveis e Condicionais

Dentre as estruturas de controle de fluxo de Python identificamos escapes e exceções, cuja respectiva sintaxe e estrutura podem ser objervadas na Figura 16. Note também, na mesma figura, que Python apresenta uma hierarquia de classes de exceção muito útil para a clareza e robustes dos cóigos escritos na linguagem. Ressaltamos também que: as construções *break* e *continue* são utilizadas nos iteráveis da linguagem, enquanto *return* é utilizado como um escape para funções e *pass* é utilizado com um escape para diversas estruturas (classes, metodos, comandos iteraveis e comandos condicionais por exemplo), veja a Figura 17 (PYTHON SOFTWARE FUNDATION, 2023b).

As estruturas iteráveis e condicionais, por sua vez, estão apresentadas na Figura 18. Observe a presença de um *else* nas estruturas iteráveis, que não é comum dentre as demais linguagens de programação. Essa instrução permite que o programador estabeleça um trecho de código para ser executado caso o iterável seja concluido, ou seja, caso o iterável não seja interrompido durante a execução. Finalmente observe também a presença de uma palavra *elif* utilizada na instrução condicional. Essa palavra é uma abreviação para o *else if* comúnmente visto na linguagem C (PYTHON SOFTWARE FUNDATION, 2023b).

Escapes

```
break   continue   pass   return
```

Exceções

BaseException
- BaseExceptionGroup
- GeneratorExit
- KeyboardInterrupt
- SystemExit
- Exception

Figura 16 – Escapes e Exceções de Python (PYTHON SOFTWARE FUNDATION, 2023b)

```python
for val in sequencia:
    if condicional: # caso a condição seja satisfeita
        break # o iterável é parado abruptamente

for val in sequencia:
    if condicional: # caso a condição seja satisfeita
        continue # o iterável pula para a próxima iteração

def funcao(): # retorna o valor obtido quando avaliada a expressão
    return expressao

class Human: # pass indica a presença de um código futuro
    pass # outra sintaxe aceita é: '...'
```

Figura 17 – Exemplos de escapes em Python (confeccionado pelos próprios autores)

```python
# Iteráveis

while expressao: ...
else: ...

for identificador in interavel: ...
else: ...

# Condicional

if expressao: ...
elif expressao: ... # opicional
else: ... # opicional
```

Figura 18 – Iteráveis e Condicionais em Python (PYTHON SOFTWARE FUNDATION, 2023b)

5 CONSIDERAÇÕES FINAIS

5.1 Bruno Oliveira Santiago

Uma das características mais notáveis do Python é sua sintaxe simples e legível. A linguagem foi projetada para ser fácil de entender e escrever, com uma sintaxe que se assemelha à linguagem natural. Isso torna o código Python altamente legível e de fácil manutenção, facilitando o trabalho em equipe e a colaboração em projetos complexos.

Outro aspecto que torna Python marcante é sua ampla biblioteca padrão. Python possui uma rica coleção de módulos e pacotes que abrangem uma ampla gama de funcionalidades, desde manipulação de strings e acesso a bancos de dados até processamento de imagens e criação de interfaces gráficas. Essa biblioteca padrão abrangente permite que os desenvolvedores realizem tarefas comuns de programação sem a necessidade de procurar bibliotecas externas, economizando tempo e esforço.

Python também é uma linguagem altamente versátil. Ela pode ser usada para desenvolver uma variedade de aplicativos, desde scripts simples até aplicativos web complexos e até mesmo projetos científicos. Sua flexibilidade torna Python uma escolha popular para iniciantes em programação, bem como para profissionais experientes em várias áreas.

Outra vantagem notável do Python é sua portabilidade. Os programas escritos em Python podem ser executados em várias plataformas, incluindo Windows, macOS e Linux, com poucas ou nenhuma modificação necessária. Isso significa que os desenvolvedores podem criar aplicativos que funcionam em diferentes sistemas operacionais, alcançando um público mais amplo.

Outra característica marcante é a quantidade de paradigmas que a linguagem possui. Por apresentar paradigmas declarativos e imperativos, Python permite que a estrutura do código se adéque ao projeto da equipe e não o contrário. Dessa forma, a flexibilidade da linguagem permite focar na solução do problema, sem precisar se esforçar em seguir uma estrutura sintática rígida.

Em resumo, Python se destaca por sua sintaxe simples e legível, sua ampla biblioteca padrão, sua versatilidade e sua portabilidade. Essas características combinadas tornam Python uma linguagem poderosa e acessível, adequada para iniciantes e especialistas em programação. Não é de surpreender que Python tenha conquistado uma posição de destaque no mundo da programação e continue a ser uma escolha preferida para uma variedade de aplicações.

5.2 Fábio Freire Kochem

O fato da linguagem Python ser interpretada é algo muito interessante, visto que não é algo característico da maioria das linguagens, tendo como exemplo mais recente o JavaScript ou PHP. Entretanto, ainda é uma dúvida para a maioria dos programadores o significado da utilização de um interpretador e da independência de um compilador, mesmo que seja um mecanismo muito interessante de se entender.

A função do interpretador Python é pegar o código-fonte escrito na linguagem e ler o mesmo, linha por linha, executando-o em sequência. A tradução das instruções do código é feita para a linguagem de máquina em tempo real, não sendo necessário compilar previamente aquele mesmo código-fonte lido pelo interpretador. Embora a linguagem seja interpretada, ela também possui um compilador, onde código-fonte é traduzido para um formato chamado bytecode Python, que é uma representação de baixo nível das instruções na linguagem. O bytecode é salvo em arquivos com a extensão '.pyc' para um uso futuro.

Agora que é possível entender as duas funcionalidades, a forma como Python as trata deixa mais interessante a relação. Durante a execução do código Python, o interpretador Python pode otimizar o desempenho do programa convertendo partes do bytecode em código de máquina, sendo esse processo conhecido com Just-in-Time, melhorando significativamente a execução do programa.

Contudo, o interpretador é extremamente necessário para executar o código, mas também o compilador pode ser uma ferramenta opcional caso seja necessário/desejado otimizar o desempenho e fornecer o bytecode compilado, para o interpretador utilizar futuramente.

5.3 Fernando Campos Silva Dal' Maria

Dentre todas as funcionalidade e principais características de Python, saliento aqui aquelas relacionadas com a implementação de equivalência de tipos e aquelas relacionadas à *List Comprehension* e ao "operador ternário"de Python. Contemplo inicialmente a peculiar equivalência de tipos da linguagem: A implementação de uma equivalência de tipos utilizando o conceito de *Duck Typing*, como já apresentado neste texto, é uma característica marcante que permite tanto o polimorfismo quanto a implementação de uma linguagem fortemente e dinamicamente tipada. Seu emprego na linguagem permite uma maior legibilidade para o código embora, caso utilizada de forma descuidada gere erros de lógica que podem ser difíceis de identificar (PYTHON SOFTWARE FUNDATION, 2023b).

Com relação as *List Comprehension* e ao "operador ternário"de Python é importante ressaltar que elas advém de outra linguagem de programação chamada Haskell, que inspirou os desenvolvedores de Python a implementá-las na sintaxe da linguagem. Ambas as construções advém do paradigma de programação funcional e são avaliadas para gerar um valor. No caso das *List Comprehension*, o valor gerado é um vetor de 'n' elementos gerados a partir de uma estrutura iterável, já no caso do "operador ternário"temos uma expressão que quando avaliada retorna um valor específico baseado em uma condição estabelecida pelo programador (PYTHON SOFTWARE FUNDATION, 2023b). Veja na Figura 19 um exemplo de uma *List Comprehension* e de uma expressão condicional.

Por fim gostaria de ressaltar, como exemplificado pelos parágrafos acima, que mesmo apresentando uma estrutura tão complexa, a linguagem é conhecida por ser de fácil aprendizagem, por apresentar uma sintaxe simples e por ser muito legível (PYTHON SOFTWARE FUNDATION, 2023b). Essa talvez, seja a característica mais chamativa de Python, já que ela permite que a linguagem seja de fácil aprendizado e possa ser aplicada em diversas instituições acadêmicas ao redor do mundo, das quais podemos citar Stanford, a Universidade Federal de Minas Gerais e a própria Pontifícia Universidade Católica de Minas Gerais.

```
# List Comprehension
identificador for identificador in interavel if expressao
```

```
# "Operador Ternário"
... if expressao else: ...
```

Figura 19 – List Comprehension e Expressão condicional semelhante a um "operador ternário"(PYTHON SOFTWARE FUNDATION, 2023b)

REFERÊNCIAS

BAIRAGI, Ishan Adhikari et al. Uno: A web application using django. In: **2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)**. [S.l.: s.n.], 2021. p. 1371–1374.

CHALLAPALLI, Sai Sri Nandan et al. Web development and performance comparison of web development technologies in node.js and python. In: **2021 International Conference on Technological Advancements and Innovations (ICTAI)**. [S.l.: s.n.], 2021. p. 303–307.

DAYLEY, Brad. **Python Phrasebook: Essential Code and Commands**. 1. ed. [S.l.]: Sams, 2006. 288 p.

DJANGO. **Django The web framework for perfectionists with deadlines.** 2023. Acesso em: 20 Maio 2023. Disponível em: <https://www.djangoproject.com/>.

DYER, Robert; CHAUHAN, Jigyasa. An exploratory study on the predominant programming paradigms in python code. In: . New York, NY, USA: Association for Computing Machinery, 2022. (ESEC/FSE 2022), p. 684–695. ISBN 9781450394130. Disponível em: <https://doi.org/10.1145/3540250.3549158>.

FLASK. **Flask User's Guide**. 2023. Acesso em: 20 Maio 2023. Disponível em: <https://flask.palletsprojects.com/en/2.3.x/>.

LUTZ, Mark. **Learning Python**. 5. ed. [S.l.]: O'Reilly, 2013. 1540 p.

MILOJKOVIC, Nevena; GHAFARI, Mohammad; NIERSTRASZ, Oscar. It's duck (typing) season! In: **2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)**. [S.l.: s.n.], 2017. p. 312–315.

NAGPAL, Abhinav; GABRANI, Goldie. Python for data analytics, scientific and technical applications. In: **2019 Amity International Conference on Artificial Intelligence (AICAI)**. [S.l.: s.n.], 2019. p. 140–145.

PYTHON SOFTWARE FUNDATION. **Applications for Python**. 2023. Acesso em: 23 Abril 2023. Disponível em: <https://www.python.org/about/apps/>.

PYTHON SOFTWARE FUNDATION. **Python 3.11.3 documentation**. 2023. Acesso em: 23 Abril 2023. Disponível em: <https://docs.python.org/3.11/>.

PYTHON SOFTWARE FUNDATION. **Python Logo**. 2023. Acesso em: 23 Abril 2023. Disponível em: <https://www.python.org/community/logos/>.

PYTHON WIKI. **The Python Wiki**. 2023. Acesso em: 20 Maio 2023. Disponível em: <https://wiki.python.org/moin/>.

ROSSUM, Guido van. **Foreword for "Programming Python"(1st ed.)**. Reston, Virginia: [s.n.], 1996. Acesso em: 23 Abril 2023. Disponível em: <https://legacy.python.org/doc/essays/foreword/>.

ROSSUM, Guido van. **Guido van Rossum - Resume**. 2023. Acesso em: 23 Abril 2023. Disponível em: <https://gvanrossum.github.io//Resume.html>.

SEBESTA, Robert W. **Concepts of programming languages**. 12. ed. [S.l.]: Pearson, 2019. 752 p.

SEVERANCE, Charles. Guido van rossum: The early years of python. **Computer**, v. 48, n. 2, p. 7–9, 2015.

SEVERANCE, Charles. Guido van rossum: The modern era of python. **Computer**, v. 48, n. 3, p. 8–10, 2015.

STACKOVERFLOW. **Developer Servey 2022**. 2022. Acesso em: 18 Maio 2023. Disponível em: <https://survey.stackoverflow.co/2022/#most-popular-technologies-language>.

VOGEL, Patrick et al. A low-effort analytics platform for visualizing evolving flask-based python web services. In: **2017 IEEE Working Conference on Software Visualization (VISSOFT)**. [S.l.: s.n.], 2017. p. 109–113.

WANG, Xudong; ZHANG, Zhimei. Analysis of the design of several modern programming languages. In: **2022 IEEE 2nd International Conference on Computer Systems (ICCS)**. [S.l.]: International Conference on Computer Systems (ICCS), 2022. p. 40–44.