

SOBRE OS AUTORES

ELIABEL BARRETO

Formado em Tecnologia em Mecatrônica Industrial pela Faculdade de Tecnologia Termomecânica em 2012, trabalhou na indústria por 10 anos e lecionou sobre diversos assuntos em faculdades e escolas técnicas com as áreas de programação com Ladder, C, Python, Kivy, SQL, HTML, CSS, JavaScript, além de ter conhecimento prático de redes industriais para os mais diversos tipos de equipamentos, como CLP's, IHM, Inversores de frequência, servo motores, supervisórios, aplicativos mobile e WEB, e confecção de dispositivos eletrônicos. Atualmente é Instrutor de Formação Profissional no SENAI - Departamento Regional de São Paulo, atuando na Escola "Almirante Tamandaré" nos cursos de ensino técnico e de formação inicial e continuada.



KÁTIA RODRIGUES

Graduada em Matemática e Pós-graduada em Gestão Empresarial pela Universidade Santa Cecília. Pós-graduanda em Indústria 4.0: "Engenharia de implementação das tecnologias habilitadoras" na Faculdade SENAI de Tecnologia Mecatrônica. Possui uma visão abrangente sobre as práticas de gestão e a integração entre tecnologia e negócios. Experiência como docente nos Cursos de Formação Inicial e Continuada e de Matemática. Atualmente é Instrutora de Formação Profissional no SENAI - Departamento Regional de São Paulo, atuando na Escola "Almirante Tamandaré" no ensino de AutoCAD, Excel Básico, Avançado, VBA e Computação em Nuvem com ênfase nas plataformas Microsoft, Google e AWS.

LITERATURA RECOMENDADA

Introdução A Python Com Aplicações De Sistemas Operacionais, Fábio

Augusto Procópio De Paiva

Learn Python The Hard Way, Zed Shaw

Pense em Python, Allen B. Downey

Aprendendo Python, Mark Lutz & David Ascher

Learning Python, Mark Lutz

1 COMO APRENDER PYTHON

1.1 O QUE REALMENTE IMPORTA?

É importante destacar que todo o conteúdo apresentado nesta apostila e durante o curso não é de autoria exclusiva de nenhum indivíduo. É fundamental observar que todos os materiais de estudo abordados ao longo do curso podem ser encontrados na internet, em forma de livros, blogs, vídeos e outras fontes.

No preparo deste material e na seleção das atividades do curso, estamos focados em fornecer uma abordagem didática para a criação de conhecimento de maneira sólida e eficaz com o objetivo de ajudá-lo a maximizar seu aprendizado e a fixar todo o conteúdo de forma clara.

1.2 SOBRE OS EXERCÍCIOS

Os exercícios serão disponibilizados no decorrer da semana. Para todo o Capítulo haverá uma lista de exercícios que precisará ser entregue para avaliação pelo Google Classroom (****exceto o capítulo atual e o capítulo 2 😊**).

A avaliação do curso consiste na entrega de todos os exercícios, de forma semanalmente.

O curso foi construído para ter 80 horas, dividido em 10 semanas, assim, haverá 10 listas de exercícios que precisará ser executada e entregue.

1.3 DÚVIDAS

Todas as dúvidas precisam ser sanadas imediatamente, lembre-se que por vezes a mesma dúvida que você possui pode ser a mesma dúvida de outros colegas na sala. Lembre-se:

“Não existe perguntas idiotas, o idiota é não perguntar nada, e se privar de um conhecimento que poderia lhe ser útil em algum momento, aprender com tudo e com todos é o que nos define.” André Menezes

2 INTRODUÇÃO AO PYTHON

2.1 BREVE HISTÓRIA

Guido Van Rossum criou o Python em 1989. Ele trabalhava no Centrum Voor Wiskunde en Informatica no início dos anos 1980, e seu trabalho era implementar a linguagem de programação conhecida como ABC.

Durante o final dos anos 1980, enquanto ainda estava no CWI, ele começou a procurar por uma linguagem de script que tivesse sintaxe semelhante ao ABC, mas que tivesse acesso às chamadas de sistema do Amoeba. Após procurar e não encontrar nenhuma linguagem que atendesse às suas necessidades, Rossum decidiu projetar uma linguagem de script simples que pudesse superar as inadequações do ABC. No final da década de 1980, Rossum começou a desenvolver o novo script, e em 1991, lançou a versão de abertura da linguagem de programação. Esta primeira versão tinha um sistema de módulo Modula-3, linguagem que foi posteriormente denominada “Python”.

Na comunidade Python, ele é conhecido como Benevolent Dictator for Life (BDFL), o que significa que ele continua a supervisionar o processo de desenvolvimento do Python, tomando decisões quando necessário.

CURIOSIDADES:

Muitas pessoas costumam pensar que o nome Python tem origem em um tipo de cobra, já que o logotipo do Python mostra a imagem de uma cobra azul e amarela.



No entanto, a verdadeira história por trás da nomenclatura é um pouco diferente. ...

Na década de 1970, a BBC tinha um programa de TV popular do qual van Rossum era um grande fã chamado Fly Circus de Monty Python, ou apenas Monty Python para os íntimos.

Assim, quando desenvolveu a linguagem, ele pensou que precisava de um nome que fosse curto, único e um pouco misterioso, e por algum motivo que só ele conhecia, decidiu chamar o projeto de "**Python**".

2.2 CONTEXTO

A linguagem Python foi uma tendência em 2019 e ainda continua em alta no mercado. Afinal, por conta da sua versatilidade, esse tipo de código oferece muitas oportunidades no processo de desenvolvimento digital.

As principais vantagens para se escolher a linguagem Python do ponto de vista de um programador são:

Aprendizado acessível:

- Python é uma linguagem gratuita e de código aberto, cujas bibliotecas e módulos são reutilizáveis;
- Desenvolvedores Python têm à disposição diversos materiais didáticos, como a sua documentação oficial e registros de outros programadores;
- As bibliotecas são gratuitas e com tradução voluntária dos códigos realizada pelos membros da comunidade, facilitando ainda mais o acesso à programação.

Sintaxe simplificada

- Python é uma linguagem simples, fácil e intuitiva, o que torna seu aprendizado muito mais rápido e dinâmico na comparação com outras linguagens de programação, como o Java ou o C++;
- A quantidade reduzida de códigos no desenvolvimento de cada projeto aumenta a produtividade e diminui a margem de erro.

Versatilidade

- A linguagem Python roda em diversas plataformas, sistemas operacionais e processadores, como Windows, MacOS, Android, Linux;
- Serve para softwares em qualquer tipo de ambiente ou interface, seja web, desktop ou dispositivo móvel;
- Cumpre bem funções bastante diversas, da ciência de dados à aplicação web e do reconhecimento facial a um software básico de finanças.

Agora vamos analisar as vantagens de se aprender a linguagem Python do ponto de vista das empresas.

Produtividade sem erro

- Python é uma linguagem simples, intuitiva e curta, o que, consequentemente, gera maior produtividade, agilidade e qualidade aos projetos;
- A sintaxe enxuta permite que a programação transcorra com menos erros e dúvidas, mesmo que feita em diversas camadas e em projetos complexos envolvendo diferentes colaboradores.

Ciência de dados

Com a ciência de dados, as empresas podem entender e analisar melhor alguns fatores:

- Avaliar resultados em determinados períodos de tempo;
- Analisar as preferências de seus clientes;
- Prever comportamentos de consumidores;
- Favorecer o relacionamento com os clientes;
- Facilitar a conquista de novos leads;
- Reduzir custos;
- Otimizar a produção.

Mas nem tudo são flores, não é mesmo; existem sim algumas desvantagens no uso do Python, e alguns programadores tendem a criticar a linguagem. Abaixo citamos alguns motivos:

- Pelo fato do Python ser uma linguagem de alto nível e muito próxima do olhar humano, alguns desenvolvedores que iniciaram seus estudos com Python tem dificuldades de migrar para outras linguagens;
- Aplicações em Python não são rápidas;
- A sintaxe é muito rígida;
- Não é recomendado para desenvolvimento mobile, pois não tem bibliotecas nativas para iOS e Android;
- Experiência limitada de programação.

2.2.1 POR QUE ESCOLHER O PYTHON?

Como vimos anteriormente, a linguagem Python possui muitas vantagens e desvantagens do ponto de vista técnico e filosófico da linguagem, como veremos no decorrer deste capítulo, mas vamos falar sobre números e onde ele é aplicado.

O Python pode ser utilizado em diversos tipos de aplicação:

Desenvolvimento Web do lado do servidor: O desenvolvimento Web do lado do servidor inclui as funções de backend complexas executadas pelos sites para exibir informações ao usuário. Por exemplo, os sites devem interagir com bancos de dados, interagir com outros sites e proteger os dados ao enviá-los pela rede.

O Python é útil para escrever código do lado do servidor, porque oferece muitas bibliotecas e frameworks, como Flask e Django, que consistem em código pré-escrito para funções complexas de backend. Os desenvolvedores também usam uma ampla variedade de frameworks Python que fornecem todas as ferramentas necessárias para criar aplicações da Web com mais rapidez e facilidade. Por exemplo, os desenvolvedores podem criar o esqueleto da aplicação Web em segundos, porque não precisam escrever do zero. Eles podem testar usando as ferramentas de testes do framework sem depender de ferramentas externas.

Automação com scripts Python: Uma linguagem de script é uma linguagem de programação que automatiza tarefas que humanos normalmente executam. Os programadores usam amplamente scripts Python para automatizar muitas tarefas do dia a dia, como as seguintes:

- Renomear um grande número de arquivos de uma só vez;
- Converter um arquivo em outro tipo de arquivo;
- Remover palavras duplicadas em um arquivo de texto;
- Executar operações matemáticas básicas;
- Enviar mensagens de e-mail;
- Baixar conteúdo;
- Executar análise de log básica;
- Encontrar erros em vários arquivos.

Ciência de dados e Machine Learning: A ciência de dados consiste em extrair conhecimento relevante a partir dos dados, enquanto o aprendizado de

máquina (Machine Learning - ML) capacita os computadores a aprenderem automaticamente com esses dados e realizar previsões precisas. Os cientistas de dados usam o Python para tarefas de ciência de dados como as seguintes:

- Corrigir e remover dados incorretos, o que é conhecido como limpeza de dados;
- Extrair e selecionar vários recursos de dados;
- Rotulagem de dados, que é a adição de nomes significativos aos dados;
- Encontrar estatísticas diferentes com base nos dados;
- Visualizar dados usando tabelas e gráficos, como gráficos de linhas, gráficos de barras, histogramas e gráficos de pizza.

Os cientistas de dados usam bibliotecas Python de ML, como Matplotlib, Pandas, NumPy, Keras e OpenCV, para treinar modelos de ML e criar classificadores que categorizam dados com precisão. Pessoas em diferentes campos usam classificadores baseados em Python para realizar tarefas de classificação, como classificação de imagens, textos e tráfego de rede; reconhecimento de fala; e reconhecimento facial. Os cientistas de dados também usam o Python para aprendizado profundo, uma técnica avançada de ML.

Desenvolvimento de software: Os desenvolvedores de software geralmente usam o Python para diferentes tarefas de desenvolvimento e aplicações de software, como as seguintes:

- Manutenção do controle de erros no código do software;
- Construção automática do software;
- Gerenciamento de projetos de software;
- Desenvolvimento de protótipos de software;
- Desenvolvimento de aplicações de desktop usando bibliotecas de interface gráfica do usuário (GUI, como o Tkinter);
- Desenvolvimento de jogos simples baseados em texto para videogames mais complexos.

Automação de testes de software: O teste de software é o processo de verificar se os resultados reais do software correspondem aos resultados esperados para garantir que o software esteja livre de erros.

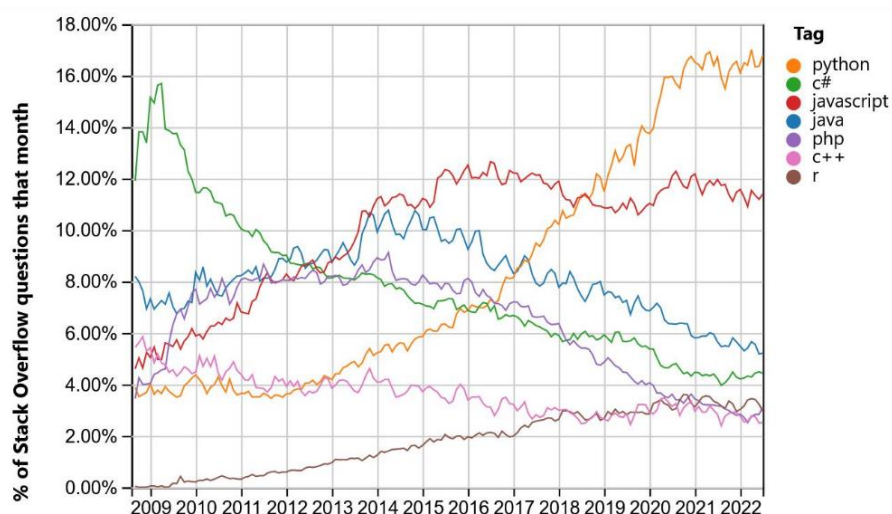
Os desenvolvedores usam frameworks de testes de unidade Python, como unittest, Robot e PyUnit, para testar as funções que escrevem.

Os testadores de software usam o Python para escrever casos de teste para vários cenários de teste. Por exemplo, para testar a interface do usuário de uma aplicação Web, vários componentes de software e novos recursos.

Os desenvolvedores podem usar várias ferramentas para executar scripts de teste automaticamente. Essas ferramentas são conhecidas como ferramentas de continuous integration/continuous deployment (CI/CD – integração contínua/implantação contínua). Testadores e desenvolvedores de software usam ferramentas de CI/CD, como Travis CI e Jenkins, para automatizar testes. A ferramenta de CI/CD executa automaticamente os scripts de teste do Python e relata os resultados do teste sempre que os desenvolvedores acrescentam novas alterações no código.

Todas essas aplicações e versatilidades que a linguagem Python traz fez com que muitas empresas adotassem em seus sistemas de trabalho. Empresas como Uber, GoldmanSachs, Netflix e Google lideram o uso dessas linguagens de programação no mercado.

Por todos esses fatores, Python é uma das linguagens com a comunidade mais ativa em fóruns online voltados à programação e é a linguagem que mais cresce no mundo:



Com certeza, após essas informações fica visível alguns motivos para se aprender a usar essa linguagem, mas podemos ser mais persuasivos se analisarmos os salários que são pagos no mercado de forma geral.

Principais 10 locais pagantes para Python no Brasil

Classificação	Localidades	Salário	Faixa salarial	Comparação	
1	Fortaleza, CE	R\$ 5.342	R\$ 5.342 - R\$ 5.342	+77%	Mostrar vagas >
2	Home office	R\$ 5.297	R\$ 4.416 - R\$ 5.536	+76%	Mostrar vagas >
3	Porto Alegre, RS	R\$ 3.952	R\$ 3.546 - R\$ 4.317	+31%	Mostrar vagas >
4	Brasília, DF	R\$ 3.553	R\$ 3.220 - R\$ 3.886	+18%	Mostrar vagas >
5	Curitiba, PR	R\$ 3.490	R\$ 2.341 - R\$ 3.490	+16%	Mostrar vagas >
6	Rio de Janeiro, RJ	R\$ 3.024	R\$ 2.667 - R\$ 3.864	0%	Mostrar vagas >
7	Barueri, SP	R\$ 2.750	R\$ 1.359 - R\$ 4.774	-9%	Mostrar vagas >
8	São Paulo, SP	R\$ 2.375	R\$ 1.382 - R\$ 5.018	-21%	Mostrar vagas >
9	Belo Horizonte, MG	R\$ 2.351	R\$ 1.149 - R\$ 2.351	-22%	Mostrar vagas >
10	Campinas, SP	R\$ 2.051	R\$ 1.661 - R\$ 2.051	-32%	Mostrar vagas >

Extraído de [https://br.jooble.org/salary/python/S%C3%A3o-Paulo-\(estado\)?salarySearchSource=1](https://br.jooble.org/salary/python/S%C3%A3o-Paulo-(estado)?salarySearchSource=1) em setembro de 2023

2.3 VERSÕES

A história do Python começa com Guido Van Rossum a iniciar o seu desenvolvimento em 1989 e a começar a implementá-lo em fevereiro de 1991, altura em que foi lançada a primeira versão pública: 0.9.0.

A versão 1.0 foi lançada em janeiro de 1994, a versão 2.0 foi lançada em outubro de 2000 e a versão 3.0 foi lançada em dezembro de 2008.

A primeira versão do Python já incluía classes com heranças, tratamento de exceções, funções e uma das suas características fundamentais: funcionamento modular. Isto permitiu que fosse uma linguagem muito mais limpa e acessível para pessoas com poucos conhecimentos de programação. Uma característica que permanece até hoje.

Até 2018, o desenvolvimento desta popular linguagem de programação era liderado pessoalmente por Van Rossum, mas ele decidiu afastar-se e, desde 2019, são cinco pessoas que decidem como evolui e se desenvolve Python. Um conselho que é renovado anualmente.

Quando foi lançada a primeira versão definitiva do Python, a popularidade desta nova linguagem de programação era tal que foi criado o comp.lang.python, um fórum de discussão Python que multiplicou ainda mais o seu número de utilizadores.

2.3.1 VERSÃO 1.0

Python é uma linguagem de programação que Guido Van Rossum começou a desenvolver enquanto trabalhava no CWI, e foi este centro de pesquisa que, em 1995 lançou a versão 1.2 do Python. A partir daí, desvinculado do CWI, Van Rossum tornou o código ainda mais acessível e, no ano 2000, a equipa principal de developers Python mudou-se para a BeOpen.com formando a equipa BeOpen Python Labs.

Python 1.6.1 é exatamente igual ao 1.6, mas com alguns bugs corrigidos e uma nova licença compatível com GPL.

A versão 1.6 do Python teve alguns problemas com o seu tipo de licença até que a Free Software Foundation (FSF) conseguiu mudar Python para uma licença de Software Livre, o que o passou a tornar compatível com GPL.

2.3.2 VERSÃO 2.0

Em outubro de 2000, foi publicada a segunda versão Python. Uma nova versão que incluía a geração de listas, uma das características mais importantes desta linguagem de programação.

Em 2001, foi criada a Python Software Foundation, que a partir do Python 2.1 possui todo o código, documentação e especificações da linguagem.

Além desta nova característica, esta nova versão do Python também incluiu um novo sistema, graças ao qual os programadores eram capaz de fazer referências cíclicas e, desta forma, Python poderia recolher o lixo dentro do código.

2.3.3 VERSÃO 3.0

A última grande atualização na história do Python ocorreu em 2008 com o lançamento da versão 3.0, que veio solucionar as principais falhas no design desta linguagem de programação.

Embora Python mantenha a sua filosofia, nesta última versão, esta linguagem de programação acumulou formas novas e redundantes para programar o mesmo elemento. Daí a necessidade de novas versões que eliminem esses construtores duplicados.

Python 3.0 quebra a compatibilidade com as versões anteriores da linguagem, uma vez que o código do Python 2.x não deve ser executado necessariamente sem modificações no Python 3.0.

A última atualização da versão 3 do Python foi lançada em agosto de 2023 e é a versão 3.11.5. Durante o nosso curso iremos demonstrar algumas diferenças entre as versões, mas recomendo que utilizemos nos exercícios a versão 3.11.4 que conforme será explicado na seção 2.7

2.4 LINGUAGEM ALTO NÍVEL VERSUS BAIXO NÍVEL

Uma das formas de se diferenciar os tipos de linguagens que existem no mercado atual através do nível de entendimento do código pelo programador. Assim podemos diferenciar as linguagens de alto e baixo nível como:

- A **linguagem de alto nível** se aproxima mais com a linguagem humana, ou seja, o objetivo principal dela é facilitar a maneira de programar, fazendo com que o programador se expresse de uma maneira mais simples.
- A **linguagem de baixo nível** está mais próxima da linguagem de máquina, seu objetivo é fazer com que o computador consiga executar o código de forma mais rápida e eficaz possível.

Podemos exemplificar com códigos em Python e em Assembly:

```
print("Ola mundo!")
```

[1] ✓ 0.0s Python

... Ola mundo!

```
1 lea si, string
2 call printf
3 hlt
4 string db "Ola mundo!", 0
5 printf PROC
6     mov AL, [SI]
7     cmp AL, 0
8     je pfend
9     mov AH, 0Eh
10    int 10h
11    inc SI
12    jmp printf
13 pfend:
14    ret
15 printf ENDP
```

Os dois códigos acima executam a mesma ação, na linha de comando do sistema operacional será escrito o texto “Ola mundo!”. Note que é muito mais fácil de entender que em Python do que em Assembly, mas do ponto de visto do processador, executar as instruções do Assembly é muito mais fácil e rápido do que as instruções de Python.

Geralmente temos linguagens de baixo nível como Assembly e C para executar tarefas por dispositivos que exigem alto nível de execução, como microcontroladores e dispositivos de baixo custo.

Já as linguagens de alto nível como Python, Java, C#, JavaScript e outros, são usadas para criar aplicações WEB, Mobile e/ou Desktop que tendem a facilitar a vida dos operadores.

2.5 LINGUAGEM INTERPRETADA VERSUS COMPILADA

Todo programa é um conjunto de instruções, seja um programa que efetue a soma de dois números ou que um envie uma solicitação pela internet. Compiladores e interpretadores recebem código legível por seres humanos e convertem-no para código de máquina, legível pelo computador.

Em uma linguagem compilada, a máquina de destino traduz o programa diretamente. Em uma linguagem interpretada, o código fonte não é traduzido

diretamente pela máquina de destino. Em vez disso, um programa diferente, o interpretador, lê e executa o código.

2.5.1 LINGUAGENS COMPILADAS

As linguagens compiladas são convertidas diretamente na máquina em um código de máquina que o processador pode executar. Como resultado, elas tendem a ser mais rápidas e mais eficientes em sua execução do que as linguagens interpretadas. Elas também dão ao desenvolvedor mais controle sobre alguns aspectos do hardware, como o gerenciamento da memória e o uso da CPU.

As linguagens compiladas necessitam de uma etapa de "build" (montagem) – elas precisam, primeiramente, ser compiladas manualmente. Você precisa "remontar" o programa sempre que precisar fazer uma alteração. Em nosso exemplo do molho, toda a tradução já está escrita antes de chegar até você. Se o autor original decidir usar um tipo diferente de óleo de oliva, a receita inteira precisaria ser traduzida novamente e reenviada a você.

Exemplos de linguagens compiladas puras são o C, o C++, o Erlang, o Haskell, o Rust e o Go.

2.5.2 LINGUAGENS INTERPRETADAS

Os interpretadores passam por um programa linha por linha e executam cada comando. Aqui, se o autor decidir que quer usar um tipo diferente de óleo de oliva, só precisaria remover o antigo e adicionar o novo. Seu amigo tradutor poderia informar isso a você quando a mudança acontecesse.

Linguagens interpretadas, antigamente, eram significativamente mais lentas do que as linguagens compiladas. Porém, com o desenvolvimento da compilação just-in-time, essa distância vem diminuindo.

Exemplos de linguagens interpretadas comuns são o PHP, o Ruby, o Python e o JavaScript.

2.5.3 VANTAGENS DAS LINGUAGENS COMPILADAS

Os programas compilados em código de máquina nativo tendem a ser mais rápidos que o código interpretado. Isso ocorre porque o processo de traduzir o

código em tempo de execução aumenta o tempo do processo, podendo fazer com que o programa seja, em geral, mais lento.

2.5.4 DESVANTAGENS DAS LINGUAGENS COMPILADAS

As desvantagens mais notáveis são:

- Tempo adicional necessário para concluir toda a etapa de compilação antes dos testes
- Dependência da plataforma do código binário gerado

2.6 PEP20 - THE ZEN OF PYTHON

Em agosto de 2004, o desenvolvedor Tim Peters, um dos pioneiros no uso da linguagem Python escreveu de forma sucinta quais foram os princípios que orientaram o BDFL para criar o design do Python, e como a linguagem deveria ser usada no dia a dia, se tornando um guia para os programadores de Python:

- Bonito é melhor que feio.
- Explícito é melhor que implícito.
- Simples é melhor que complexo.
- Complexo é melhor que complicado.
- Linear é melhor do que aninhado.
- Esparso é melhor que denso.
- Legibilidade conta.
- Casos especiais não são especiais o bastante para quebrar as regras.
- Ainda que praticidade vença a pureza.
- Erros nunca devem passar silenciosamente.
- A menos que sejam explicitamente silenciados.
- Diante da ambiguidade, recuse a tentação de adivinhar.
- Deveria haver um — e preferencialmente só um — modo óbvio para fazer algo.
- Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.
- Agora é melhor que nunca.
- Embora nunca frequentemente seja melhor que já.

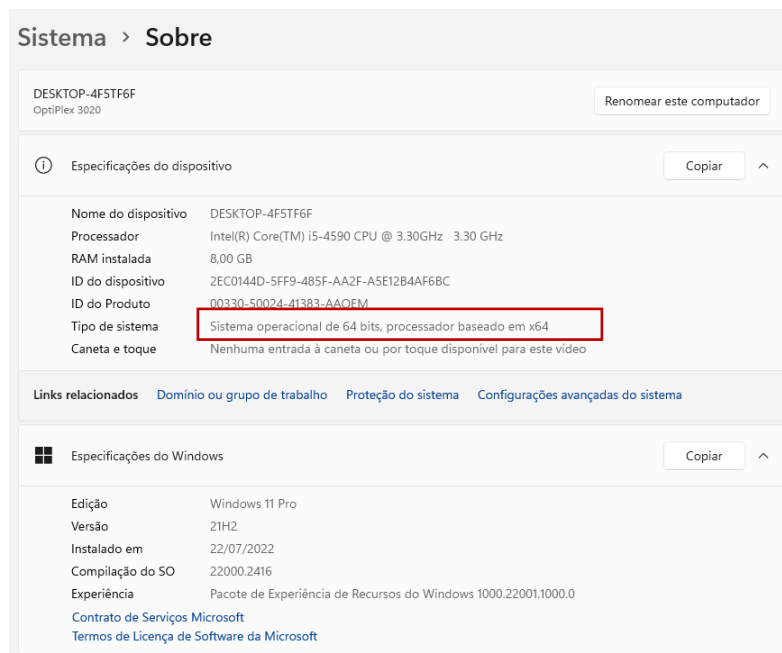
- Se a implementação é difícil de explicar, é uma má ideia.
- Se a implementação é fácil de explicar, pode ser uma boa ideia.
- Namespaces são uma grande ideia — vamos ter mais dessas!

2.7 AMBIENTE DE PROGRAMAÇÃO

Existem diversas formas de se trabalhar com o Python, no decorrer do curso iremos usar o VS Code como editor de texto padrão e o ambiente iremos usar o pacote Anaconda. Nesta documentação você vai encontrar o procedimento para instalar usando o Windows 11. Caso tenha outro sistema operacional você poderá encontrar diversos tutoriais, inclusive na documentação original de como proceder com a instalação.

2.7.1 INSTALAÇÃO DO ANACONDA

Baixar e instalar o Anaconda é fácil. O primeiro passo é descobrir se seu sistema Windows é 32-Bit ou 64-Bit. Acesse o menu iniciar do Windows 11 e pesquise por “Configurações”. Após acessar o menu de configurações, selecione a aba “Sistema” e então clique sobre a última opção, chamada “Sobre”. Uma tela como essa deve aparecer. Note que na parte Tipo de Sistema aparece justamente o tipo do seu sistema Windows, se 32-bits ou 64-bits.



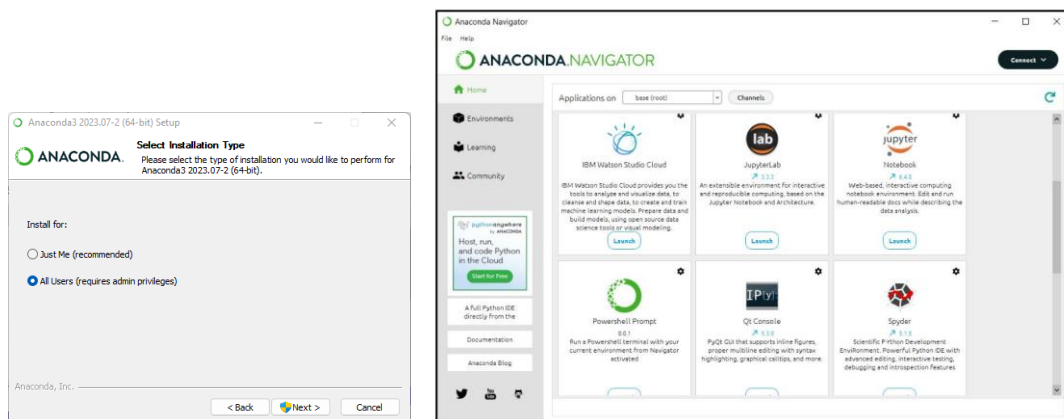
Após verificar que o sistema é 64 bits, acessar a aba de downloads no site do Anaconda (<https://www.anaconda.com/download>), em seguida, ir até o fim da

barra de rolagem da página e selecionar 64-Bit Graphical Installer do Python 3.11 para baixar o arquivo executável necessário para a instalação.



Depois de baixado, basta clicar duas vezes sobre o arquivo e ir acompanhando o instalador, mantendo sempre as opções padrão e escolhendo a opção de instalar para todos os usuários do computador, o que requer acesso de administrador.

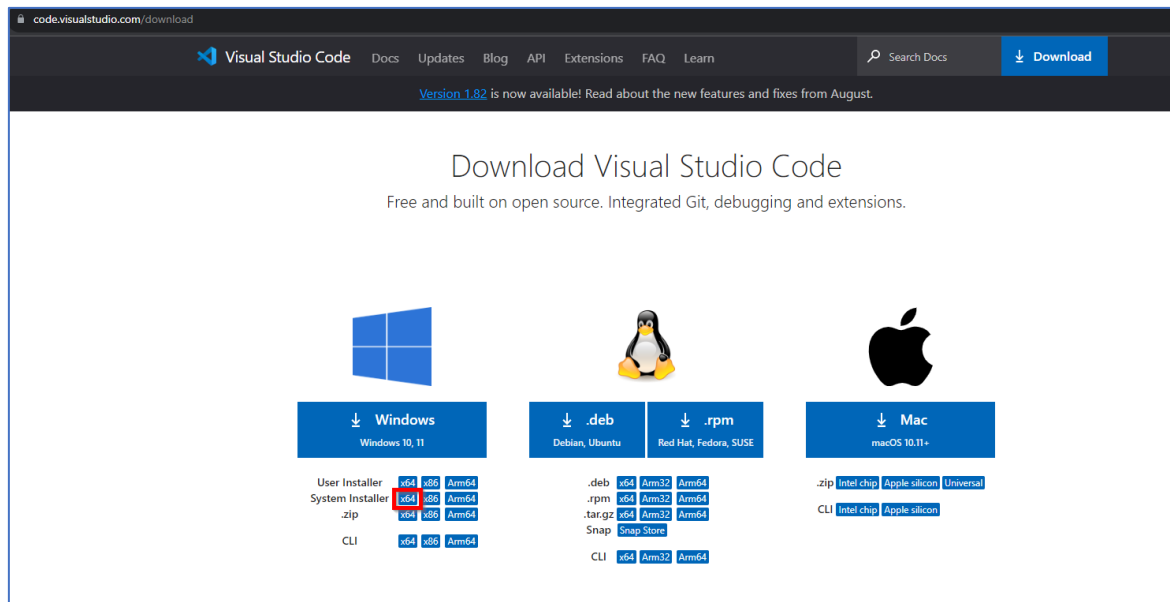
Pronto, o Anaconda está instalado e os principais pacotes e programas que utilizaremos também!



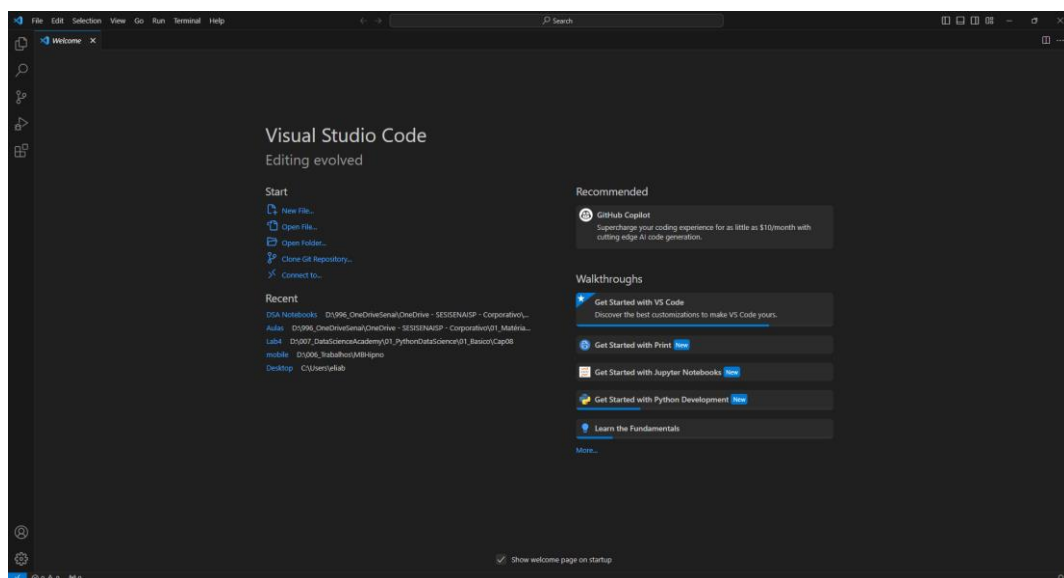
2.7.2 INSTALAÇÃO DO VS CODE

O VS Code é uma IDE (Ambiente de Desenvolvimento Integrado), uma ferramenta que reúne várias funcionalidades para auxiliar os desenvolvedores durante o processo de criação, edição, depuração e execução de código. No contexto do desenvolvimento em Python, uma boa IDE pode tornar o processo mais eficiente, produtivo e organizado.

Para realizar a instalação do VS Code iremos baixar a versão compatível com nosso sistema operacional do site oficial (<https://code.visualstudio.com/download>) e proceder com a instalação normalmente.



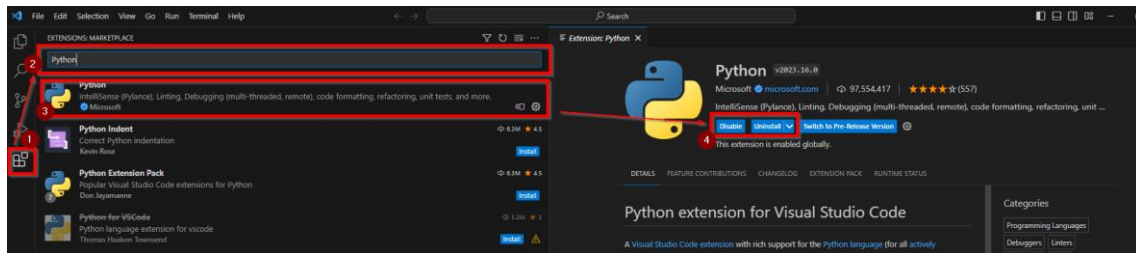
Feita a instalação podemos iniciar a aplicação e teremos a seguinte tela:



2.7.3 EXTENSÕES NO VS CODE

O VS Code é um software de código aberto mantida pela Microsoft, e há uma grande comunidade que propõe muitas melhorias para aperfeiçoar e facilitar o uso desses recursos durante o processo de programação. Essas melhorias são agrupadas e chamadas de Extensões.

Para nosso curso iremos usar uma extensão que irá ajudar muito no processo de execução do nosso código e também na parte de intellisense. Para instalar basta seguir o passo a passo abaixo:



- (1). Clicar sobre o ícone de Extensões do VS Code
- (2). Digitar o nome “Python” para localizar a extensão
- (3). Selecionar a opção “Python, Microsoft”
- (4). Clicar sobre o botão Instalar, se o seu já estiver instalado, vai aparecer como na imagem acima.

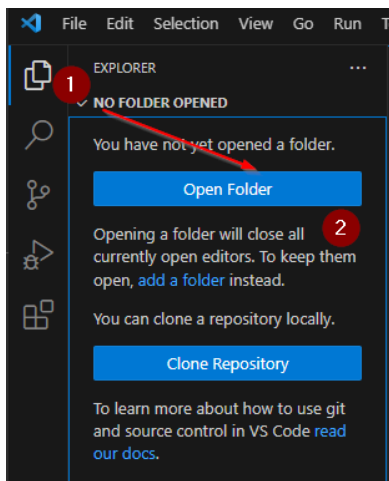
Confirme se as seguintes extensões também estão instaladas:

- **Jupyter, Microsoft:** Extensão que garante a integração com o Jupyter notebook e nos fornece ferramentas que irão facilitar o manuseio dos dados.
- **Markdown Checkboxes, Matt Bierner:** Necessário para aparecer os checkboxes dos exercícios que serão realizados nos próximos capítulos.
- **VSCODE-ICONS, VSCode Icons Team:** Deixa mais bonito e fácil de encontrar os arquivos na barra de diretório do VS Code.
- **Reload, natqe:** Vai facilitar quando Precisarmos reiniciar o VS Code para atualizar as configurações.
- **Portuguese (Brazil) Language Pack for Visual Studio, Microsoft:** Pacote de idioma português caso queira trocar os textos do VS Code.
- **Code Runner, Jun Han:** Recurso muito útil quando formos utilizar as extensões de arquivos .py
- **Rainbow Brackets, Mhammed Talhaouy:** Deixa coloridos os parênteses, mostrando as relações entre eles, vai facilitar para localizar onde começa e termina cada um.

- **Code Spell Checker, Street Side Software:** Corretor ortográfico para o idioma inglês
- **Brazilian Portuguese - Code Spell Checker, Street Side Software:** Corretor ortográfico para o idioma português (necessita que seja instalado o anterior para correto funcionamento).
- **IntelliCode, Microsoft:** Fornecerá dicas para o programador enquanto este estiver digitando o código.

2.7.4 TESTANDO O SOFTWARE PELA PRIMEIRA VEZ

Para abrir os arquivos que iremos trabalhar usaremos o Explorer do VS Code:

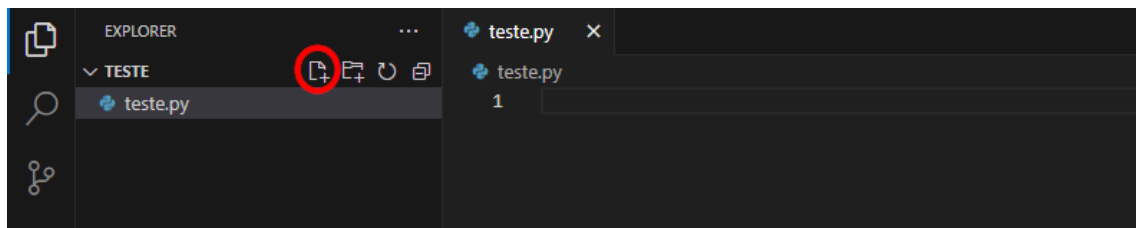


Iremos clicar no ícone da imagem (1) ao lado para abrir o navegador e criarmos uma pasta para armazenar nossos arquivos.

Irei criar uma pasta na Área de Trabalho para testar a configuração do software e demonstrar os recursos deste.

Criado a pasta e aberto a mesma dentro da IDE, vamos inicializar alguns arquivos para podermos verificar o funcionamento correto das ferramentas.

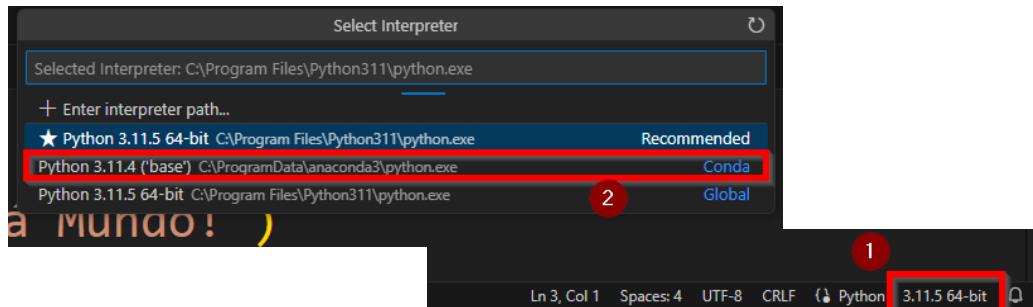
Vamos clicar sobre o ícone para criar um arquivo novo como o nome "teste.py" e clicar no ENTER. O resultado deverá ser o seguinte:



Vamos agora digitar nosso primeiro código com Python. Iremos usar a seguinte sintaxe:

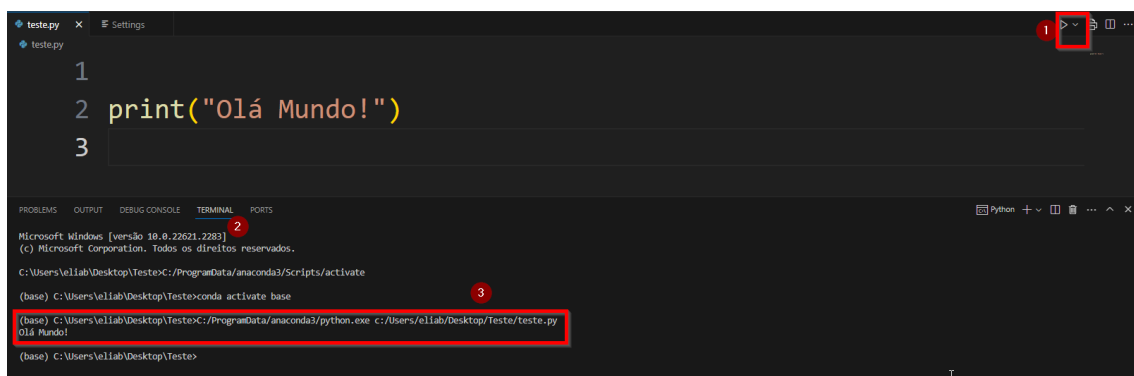
```
1
2 print("Olá Mundo!")
3
```

Tendo inserido o texto acima podemos realizar o teste de execução. Para isso precisamos verificar qual ambiente está selecionado no VS Code para rodar o código, conseguimos vendo no seguinte caminho:



Clique sobre (1) e ele deverá abrir uma lista na parte superior do software, iremos selecionar o ambiente de programação que precisamos para nossos estudos. Selecione a opção (2) *Python 3.11.4('base')* que é ambiente de programação Anaconda que foi instalado.

Feito isso podemos rodar nosso código sem problemas, como podemos ver na imagem abaixo, clicando sobre o botão Run Python File(1) uma nova aba do terminal irá abrir (2) e nosso código será executado conforme demonstrado no item (3).



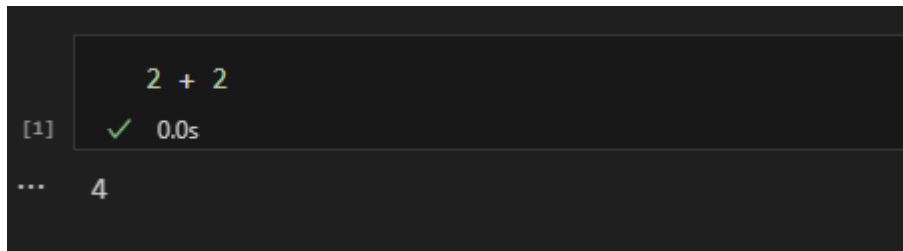
Caso os passos tenham sido executados, você terá o ambiente configurado de acordo com o necessário para nossas aplicações e estudos.

2.7.5 O QUE SÃO OS NOTEBOOKS DO PYTHON

Durante o nosso curso iremos utilizar um outro formato de arquivo que será diferente da **extensão.py**, usaremos a **extensão.ipynb** que servirá de base para os nossos estudos e também para a entrega dos exercícios.

Este arquivo que estamos criando se chama **notebook**: contém textos explicativos e códigos de programas intercalados, em um só arquivo. Em um

notebook, as unidades básicas de texto ou código são chamadas de **células**. Sendo fácil distinguir as células de código, pois elas vêm precedidas por `[]`, como na imagem abaixo.



2.7.6 MANIPULAÇÃO DE CÉLULAS

As seções abaixo apresentam uma descrição geral de como manipular as células. Os comandos exatos dependem do ambiente utilizado.

Navegação

Para navegar entre as células, basta clicar nela ou utilizar as flechas do teclado.

Edição

Para editar uma célula, selecione a mesma e aperte ENTER. Alternativamente, você pode dar um clique duplo em cima da célula.

Execução

Você pode **executar uma célula** clicando no botão **play** do lado esquerdo da célula ou pela combinação das teclas SHIFT+ENTER.

- Se for uma célula tipo código (Python), o código será interpretado.
- Se for uma célula tipo texto (Markdown), o texto será formatado.

Criação

Você pode criar uma célula com o mouse através do botão adicionar entre as células ou utilizar o menu. Ao criar uma célula, você deverá escolher o tipo (Código/Python ou Texto/Markdown).

Você também pode alterar o tipo de célula posteriormente, utilizando o menu.

Obs.: Muitos ambientes fornecem outros tipos de células, como SQL, LaTeX, dentre outros. Neste curso, só utilizaremos as células tipo **Python** (para códigos) e **Markdown** (para textos).

2.7.7 CÉLULAS TIPO MARKDOWN (TEXTO)

As células de texto utilizam a formatação **Markdown**: uma linguagem de marcação simples que converte seu texto em HTML válido.

Exemplos da sintaxe:

✓ Título

O símbolo usado para formatar um título em Markdown é a cerquilha #. Na construção do título formate o código colocando uma cerquilha no início da linha, demarcando o cabeçalho.

A quantidade de cerquilhas que você utiliza no início da linha indica o nível do título, sendo assim, a formatação **### Meu título** cria um cabeçalho de nível 3.

Markdown	Resultado
# Heading level 1	Nível de título 1
## Heading level 2	Nível de título 2
### Heading level 3	Nível de título 3
#### Heading level 4	Nível de título 4
##### Heading level 5	Nível de título 5
##### Heading level 6	Nível de título 6

✓ Parágrafos

A formatação mais simples do Markdown é a do parágrafo. Para criar um, basta separar o texto com uma linha em branco pressione a tecla **Enter** para criar fragmentações que o processador Markdown interpretará como parágrafos.

Markdown	Resultado
Eu realmente gosto de usar Markdown.	Eu realmente gosto de usar Markdown.
Acho que vou usá-lo para formatar todos os meus documentos a partir de agora.	Acho que vou usá-lo para formatar todos os meus documentos a partir de agora.

✓ Quebras de linha

Para gerar uma quebra de linha no Markdown, insira dois ou mais espaços em branco no final da linha e pressione a tecla **Enter**.

Markdown	Resultado
Esta é a primeira linha.<enter> E esta é a segunda linha.	Esta é a primeira linha. E esta é a segunda linha.

✓ Ênfase

Dentre as formatações de ênfase mais populares, estão o negrito e o itálico, que também podem ser aplicados usando as sintaxes.

○ Negrito

Para formatar um texto em negrito no Markdown, coloque dois asteriscos ****** antes do conteúdo e dois asteriscos ****** depois do conteúdo.

Markdown	Resultado
Eu simplesmente amo **texto em negrito** .	Eu simplesmente amo texto em negrito .

○ Itálico

Semelhante ao negrito, no caso do itálico, use somente um asterisco ***** antes do conteúdo e um asterisco ***** depois do conteúdo.

Markdown	Resultado
O texto em *itálico é legal* .	O texto em <i>itálico é legal</i>

Somente negrito ou somente itálico? Caso queira, você pode combinar as duas formatações para destacar qualquer conteúdo.

Para aplicar o negrito e o itálico simultaneamente adicione três asteriscos ******* ao redor do texto a destacar.

Markdown	Resultado
Este texto é *** muito importante *** .	Este texto é <i>muito importante</i> .

✓ Listas ordenadas

Vamos começar ordenando os elementos da nossa lista, para isso coloque um número seguido por um ponto e o texto do elemento.

Vale notar que, os números dos elementos não precisam estar em ordem numérica ascendente, porém, o primeiro elemento da lista deve iniciar com o número um 1.

NOTA: Para criar um elemento recuado insira uma tabulação antes do número que demarca o elemento. Em alguns processadores Markdown você também pode inserir 3 espaços antes da numeração, o resultado é semelhante, porém isso pode variar dependendo do aplicativo, na dúvida use o TAB.

Em algumas outras linguagens de marcação leves permitem que você use um parêntese como um delimitador, por exemplo: **1) Primeiro Item**, mas nem todos os aplicativos de Markdown oferecem suporte a isso, portanto, não é uma ótima opção do ponto de vista da compatibilidade. Para compatibilidade, use apenas ponto.

Markdown	Resultado
1. Primeiro item 2. Segundo item 3. Terceiro item 4. Quarto item	1. Primeiro item 2. Segundo item 3. Terceiro item 4. Quarto item
1.Primeiro item 1.Segundo item 1.Terceiro item 1. Quarto item	1. Primeiro item 2. Segundo item 3. Terceiro item 4. Quarto item

1. Primeiro item 8. Segundo item 3. Terceiro item 5. Quarto item	1. Primeiro item 2. Segundo item 3. Terceiro item 4. Quarto item
1. Primeiro item 2. Segundo item 3. Terceiro item 1. Item recuado 2. Item recuado 4. Quarto item	1. Primeiro item 2. Segundo item 3. Terceiro item 1. Item recuado 2. Item recuado 4. Quarto item

✓ Listas não ordenadas

Uma lista não ordenada é uma sequência de elementos denotados, normalmente, por algum símbolo à esquerda do elemento, isso é, não há uma sequência numérica aqui.

A marcação do Markdown para criar listas não ordenadas é o sinal de mais +, ou o traço -, ou o asterisco *. Coloque um desses dois sinais antes do item da lista para demarcá-lo, seguido por um espaço em branco e o conteúdo do elemento.

NOTA: Recue um elemento da lista inserindo uma tabulação antes da marcação do elemento.

Markdown	Resultado
- Primeiro item - Segundo item - Terceiro item - Quarto item	<ul style="list-style-type: none"> • Primeiro item • Segundo item • Terceiro item • Quarto item
* Primeiro item * Segundo item * Terceiro item * Quarto item	<ul style="list-style-type: none"> • Primeiro item • Segundo item • Terceiro item • Quarto item
+ Primeiro item + Segundo item + Terceiro item + Quarto item	<ul style="list-style-type: none"> • Primeiro item • Segundo item • Terceiro item • Quarto item

<ul style="list-style-type: none"> - Primeiro item - Segundo item - Terceiro item <ul style="list-style-type: none"> - Item recuado - Item recuado - Quarto item 	<ul style="list-style-type: none"> • Primeiro item • Segundo item • Terceiro item <ul style="list-style-type: none"> ◦ Item recuado ◦ Item recuado • Quarto item
---	---

✓ Iniciando itens de lista não ordenados com números

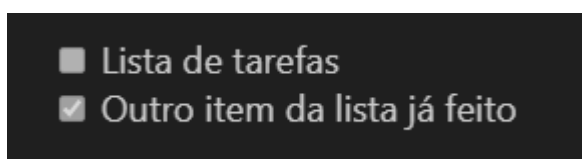
Se você precisar iniciar um item de lista não ordenado com um número seguido de um ponto, poderá usar uma barra invertida \ para escapar

Markdown	Resultado
<ul style="list-style-type: none"> - 1968\ Um grande ano! - Acho que 1969 foi o segundo melhor. 	<ul style="list-style-type: none"> • 1968 Um grande ano! • Acho que 1969 foi o segundo melhor.

✓ Listas de tarefas

Uma lista de tarefas em Markdown permite que o usuário crie uma sequência de elementos com caixas de seleção à sua esquerda usando colchetes []. Como estamos trabalhando com lista é necessário iniciar a linha do item com um traço -. Para marcar uma tarefa como concluída coloque um x dentro dos colchetes [x].

- [] Lista de tarefas
- [x] Outro item da lista já feito



✓ Tabela:

Para criar uma tabela em Markdown nós usamos traços - e barras verticais | para separar linhas e colunas.

A primeira linha da tabela é onde construímos o cabeçalho, separando essa linha por três ou mais traços --- para que o processador Markdown entenda a formatação.

A separação das colunas é feita usando a barra vertical |, também chamada de *pipe* por programadores.

| Padrão | Esquerda | Centralizado | Direita |

| --- | :--- | :---: | ---: |

| A | B | C | D |

| 1 | 2 | 3 | 4 |

Tabela:

Padrão	Esquerda	Centralizado	Direita
A	B	C	D
1	2	3	4

✓ Blockquotes

Para marcar um conteúdo como blockquote no Markdown, use o sinal de maior que > no começo do parágrafo.

> Primeiro parágrafo do blockquote

> Segundo parágrafo do blockquote



Para criar um bloco aninhado utilize dois sinais de maior que >> antes do parágrafo.

> Primeiro parágrafo do blockquote.

>

>> Esse parágrafo será interpretado como um blockquote aninhado.

✓ Conteúdo como Código

- Crases - Backticks

Usando o Markdown nós podemos demarcar uma parte do conteúdo como código usando crases `. Coloque uma crase antes e depois do texto que será interpretado pelo processador Markdown como código.

Para indicar uma palavra ou frase como código, coloque-a em backticks `

Markdown	Resultado
No prompt de comando, digite `nano`.	No prompt de comando, digite .nano

○ Escapando Crases - Backticks

Você também poderá precisa colocar uma crase dentro de uma demarcação de código sem que essa crase seja interpretada como marcação Markdown.

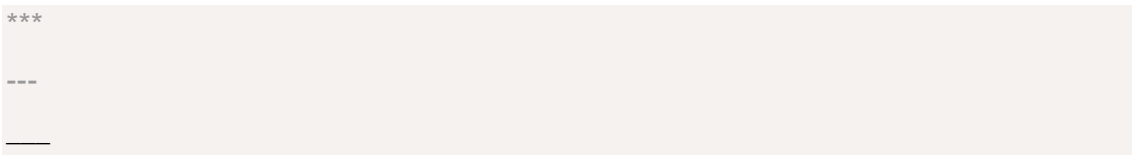
A solução é simples, duplicar a crase inicial e a crase final que cercam o pedaço de código.

Markdown	Resultado
``Use `crase dupla` em sua marcação``	Use `crase dupla` em sua marcação

✓ Linhas horizontais

Três asteriscos seguidos ***, ou três traços ---, ou três sublinhado __ criam uma linha horizontal.

Uma linha deve conter apenas esses caracteres para que o processador Markdown interprete a linha horizontal da maneira correta.



✓ Links

Links são utilizados para criar ligações entre partes do seu documento. Quando falamos em criar página para a Web ou até mesmo arquivos PDF, ser capaz de inserir links é uma das características indispensáveis para editar esses tipos de documentos.

A sintaxe básica de criação de links no Markdown é a seguinte:

- Coloque o texto que irá representar o link entre colchetes, exemplo:
`[O Melhor Site de Markdown do Mundo]`
- Em seguida, o endereço do link, também conhecido como URL, deve ser colocado entre parenteses, exemplo:
`(https://markdown.net.br)`

Juntando tudo.

Visite `[O Melhor Site de Markdown do mundo](https://markdown.net.br)`.

E a saída:

Visite [O Melhor Site de Markdown do Mundo](https://markdown.net.br)

Ao adicionar asteriscos ao redor da formatação de link, ou seja, antes dos colchetes e depois dos parenteses, você indica para o processador Markdown que aquele link deve ser enfatizado.

`**[Link *enfatizado*](https://duckduckgo.com/)**`

✓ Imagem:

A sintaxe do Markdown para adicionar uma imagem é a seguinte:

1. Um ponto de exclamação: `!`
2. O texto alternativo da imagem entre colchetes: `[]`
3. O endereço completo da imagem dentro de parênteses: `()`
4. Título opcional entre aspas, ainda dentro do parênteses: `(" ")`

Exemplos:

`![Markdown é a linguagem de marcação mais simples do mundo!](https://markdown.net.br/assets/img/markdown.jpg "Logo do Markdown")`



! [Texto

alternativo] ([https://upload.wikimedia.org/wikipedia/commons/thumb/8/8c/SENAI S%C3%A3o Paulo logo.png/1280px-SENAI S%C3%A3o Paulo logo.png](https://upload.wikimedia.org/wikipedia/commons/thumb/8/8c/SENAI_S%C3%A3o_Paulo_logo.png/1280px-SENAI_S%C3%A3o_Paulo_logo.png))



Para mais sintaxes, busque na documentação original da linguagem Markdown (<https://www.markdownguide.org/>)

2.7.8 CÉLULAS TIPO PYTHON (CÓDIGO)

As células do tipo código precisam ser elaboradas utilizando uma sintaxe específica, no caso, Python. Qualquer erro irá interromper a interpretação do código, e será apresentado uma mensagem de erro com a sua respectiva descrição.

No decorrer das próximas aulas iremos apresentar mais informações sobre a linguagem e como pode ser utilizada.

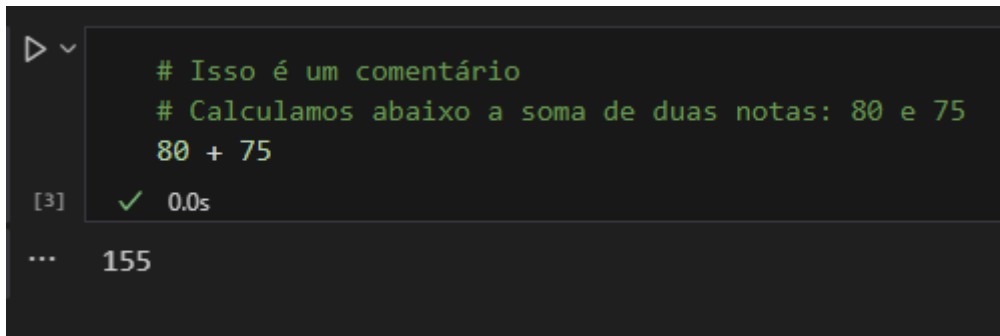
Não precisa aprender todo o conteúdo de cabeça. É importante você entender o conteúdo apresentado e depois pode consultar este Notebook quando for fazer os exercícios.

2.7.9 COMENTÁRIOS NOS CÓDIGOS

Colocar os comentários nos códigos pode ser bastante útil, para explicar o que cada linha (ou parte) do programa faz (especialmente em códigos mais longos com quais iremos lidar mais para frente de curso).

O comando `#` significa que tudo que aparece na mesma linha após este símbolo é considerado apenas como comentário (e não vai ser executado na célula de código).

Veja este exemplo:

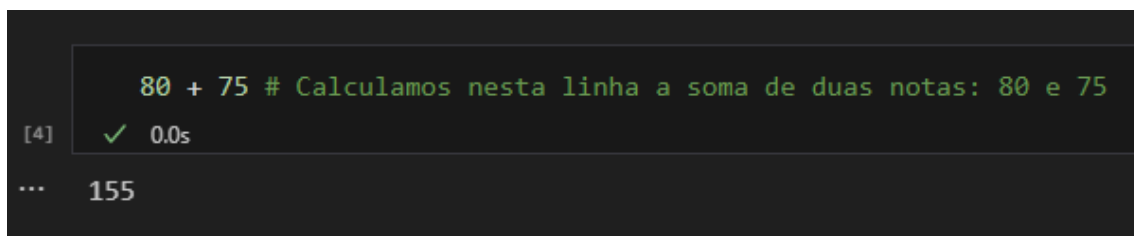


```
# Isso é um comentário
# Calculamos abaixo a soma de duas notas: 80 e 75
80 + 75
```

[3] ✓ 0.0s

... 155

Ou:



```
80 + 75 # Calculamos nesta linha a soma de duas notas: 80 e 75
```

[4] ✓ 0.0s

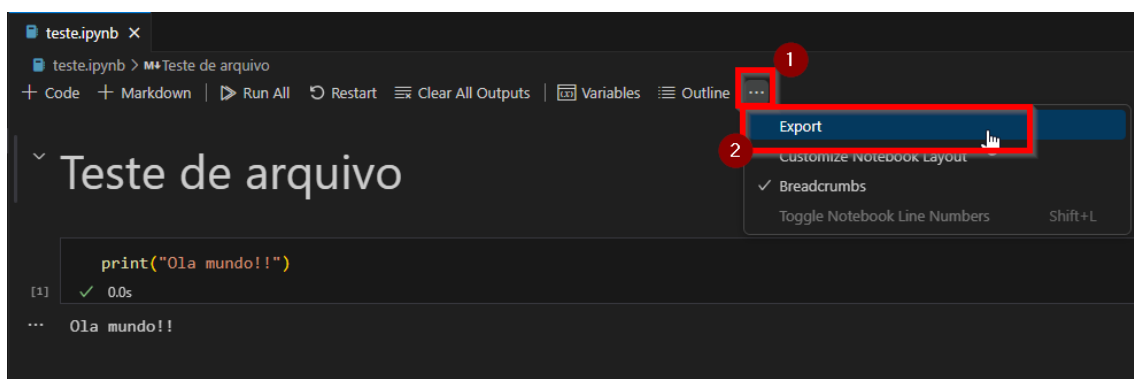
... 155

2.7.10 ENVIO DOS ARQUIVOS EM PDF

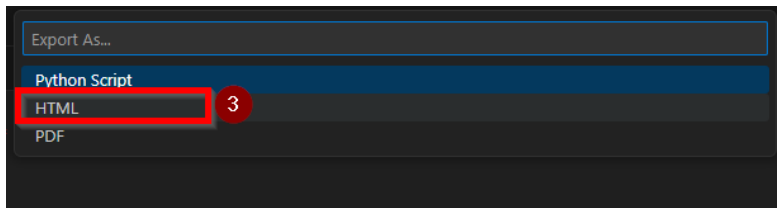
Para enviar os arquivos em PDF nas atividades do Classroom precisaremos fazer duas conversões de arquivo.

1. Primeiro iremos através do VS Code exportar em HTML

Para isso vamos no arquivo em questão e iremos exportar seguindo os passos abaixo:

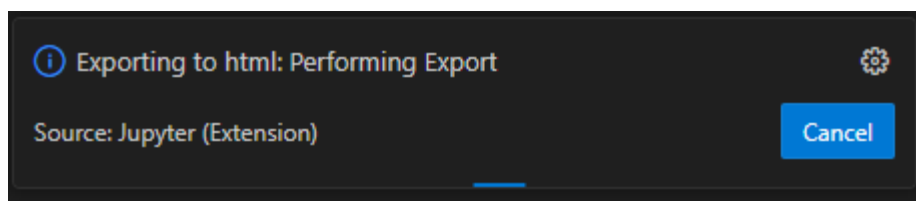


Uma caixa para selecionar a opção HTML vai abrir na parte de cima do software. (A opção PDF não funciona corretamente, por isso iremos exportar para HTML)

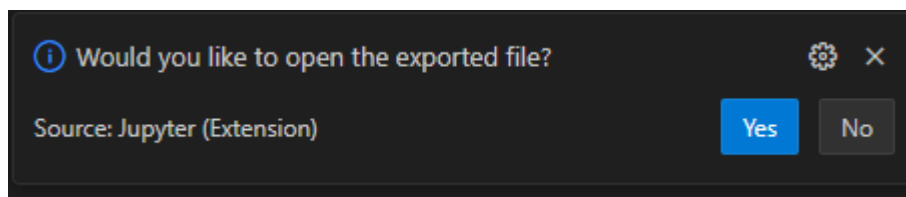


Em seguida uma janela para escolher o nome e local do arquivo irá se abrir, conforme na imagem abaixo, vou selecionar a opção Área de Trabalho e colocar o nome de teste.html

Uma caixa de diálogo irá aparecer na parte inferior direita do software, informando que o processo de exportação está em andamento.



Quando o arquivo terminar, outra caixa de diálogo irá aparecer, perguntando se deseja abrir o arquivo gerado. Iremos clicar em SIM.

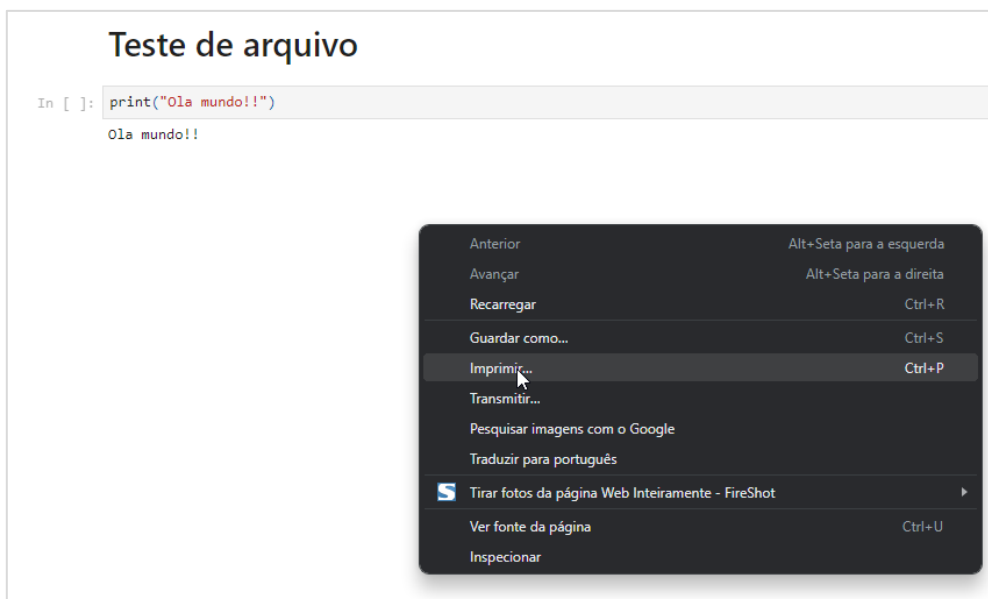


2. Através do navegador vamos exportar para PDF.

Quando abrirmos o arquivo gerado no passo anterior teremos a seguinte visualização. Pode variar de acordo com o navegador que estiver utilizando.

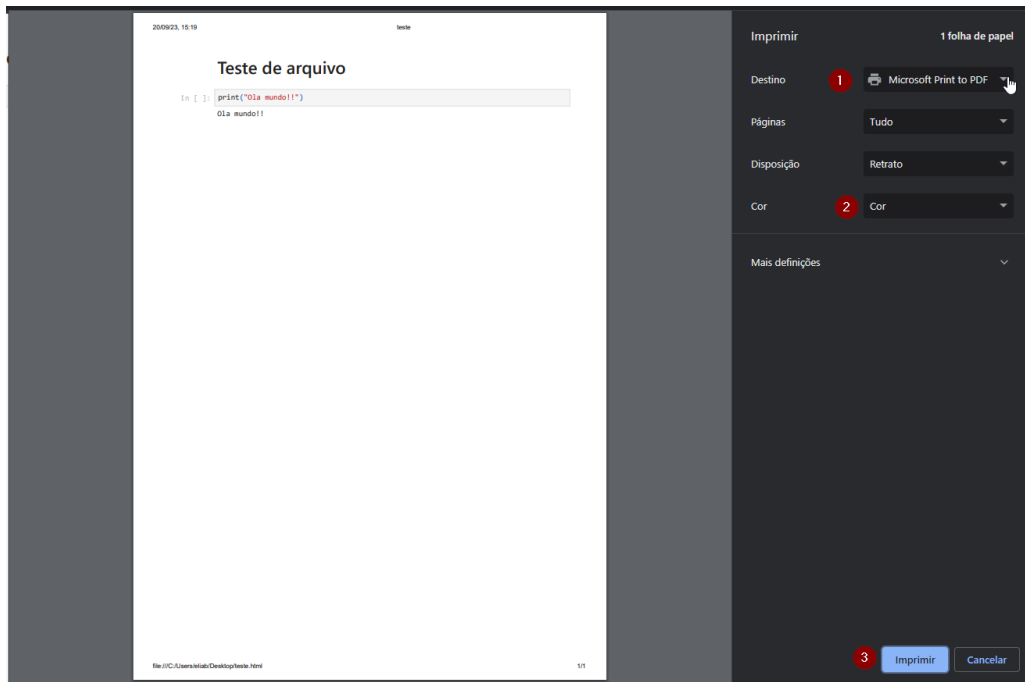


Aberto esse arquivo iremos imprimir a página. Para isso podemos clicar com o botão direito do mouse na tela ou usar o atalho **CTRL + P**.

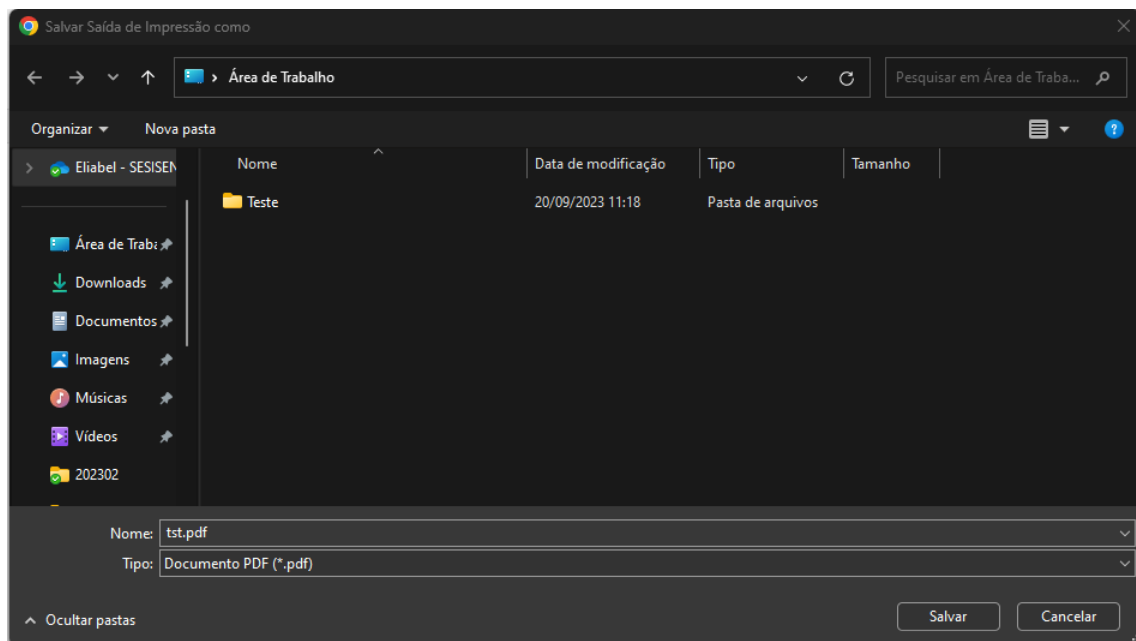


Uma janela parecida com a de baixo deve abrir, nela iremos selecionar (1) a impressora que vamos utilizar para exportar para PDF, no meu caso Microsoft Print to PDF já vem instalado no sistema operacional de fábrica.

Em (2) confirmar se a opção colorida está instalada, e ao final clicar em (3) para realizar a impressão



Novamente deverá abrir uma janela para salvar o arquivo gerado, agora deve ser um arquivo em PDF.



3. O arquivo final deverá ser enviado pelo Google Class.

Em posse do arquivo final você deverá ir até o Google Class entrar na sala do Curso, lembre-se nossa turma é PYTHON_OUT2023T. Na guia ATIVIDADES (2) você terá que achar qual a atividade que precisará ser postada. Para entrar na atividade precisará clicar no botão Conferir instruções.

Uma vez dentro da atividade você poderá ver as instruções (1), nesse campo estará todos os detalhes e arquivos de suporte para a execução da atividade. Feita a atividade você irá enviar pelo botão + Adicionar ou criar (2)

Uma vez adicionado o arquivo você deverá ter sido carregado como mostrado em (3) na tela abaixo. Concluído esse passo você deverá clicar em Entregar (4) para concluir a atividade.