PYTHON CHEAT SHEET

# GIT

## - Criar Repositório no GitHub

* <https://github.com/new>
* Preencher o nome do repositório
* exemplo de repositório criado **https://github.com/username/nome\_do\_repositorio**

## - Configurar seu github no terminal

* git config --global user.name "Seu Nome"
* git config --global user.email "[email@domain](mailto:email@domain)"

## - Clonar seu github

* Escolha a pasta em seu computador onde está o projeto
* Clone com https: **git clone https://github.com/username/nome\_do\_repositorio**
* Clone com SSH: **git clone** [**git@github.com**](mailto:git@github.com)**:username/nome\_do\_repositorio.git**

## - Comandos básicos

* **git status** para verificar o estado dos arquivos e se existem mudanças a aplicar
* **git pull** para baixar mudanças do github para o local
* **git add** **.** (para todos os arquivos) ou **git add nome\_do\_arquivo.extensao** para adicionar arquivos novos e adicionar mudanças em arquivos existentes
* **git commit -m** para o comentar as mudanças realizadas e marcar um checkpoint no histórico do repositório
* **git push** para enviar as mudanças locais para o github

# PYTHON SCRIPT

## - Shebang

O comentário especial **Shebang** especifica qual interpretador será usado para executar o programa. Desta forma é possível omitir o interpretador e executar o script diretamente pelo seu nome.

* No arquivo Python:
* No terminal:

- Permissão para executar o script:

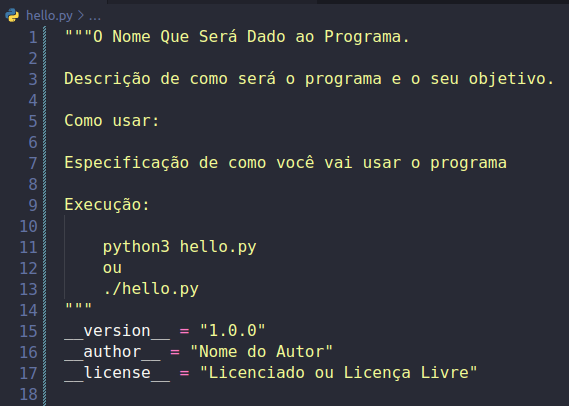
- As duas formas de executar o script:

Figura 1: Especificando o interpretador

Figura 2: Usando o Shebang

## - Docstring e metadados dunder

Em todo script Python é uma boa prática incluir um comentário de multi linhas logo nas primeiras linhas do script explicando o objetivo do script e provendo documentação para o usuário.



## - Tipos de Dados

Primários:

Os tipos primários também chamados de tipos "escalares" (scalar types) são utilizados para armazenar uma única unidade de informação como por exemplo um número ou um texto.

* **Inteiro**: O tipo usado para armazenar os números inteiros em Python é representado pela classe **int**. Ex.: idade = **25**, numero\_da\_rua = **1650**, quantidade = **346**
* **Float:** O tipo usado para armazenar os números fracionados ou decimais . Ex.: Preço = **5.90**, latitude = **-37.80467681**
* **Booleanos:** O tipo booleano é representado pela classe **bool** e ele pode armazenar apenas 2 estados: Verdadeiro e Falso (True e False) ou 1 e 0. Ex.: 1 + 1 = 2 **(Verdadeiro)**
* **NoneType:** Em alguns casos precisamos inicializar uma variável porém ainda não temos o valor para armazenar nela, nesse caso usamos o objeto **None**. Ex.: cliente = **None**
* **String:** São valores representado por letras, palavras ou conjunto de palavras e fica representado dentro de aspas (simples ou duplas). Ex.: Nome = **‘José’**

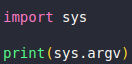
Compostos:

Com os tipos **primários** temos a limitação de representar apenas uma única informação em cada objeto, porém existem casos em que desejamos compor um objeto único que contém mais de uma informação e para isso usamos os tipos compostos.

* **Tuplas:** As tuplas representada **( )**, conseguimos fazer uma sequência de valores que podem ser de qualquer tipo. A característica importante e que talvez seja decisiva na hora de escolher usar tuplas é o fato de que elas são imutáveis, uma vez criada a tupla, não é possível alterar, não dá para mudar os valores ou adicionar novos. Ex.: compras = **(açúcar, arroz, feijão)**
* **Listas:** São bastante similares as tuplas e a maioria das operações que podemos fazer com tuplas também podemos fazer com as listas, uma das grandes diferenças está na implementação de protocolos de edição dos elementos, portanto as listas são mutáveis e permitem que incluamos novos itens, permitem a remoção de itens existentes e a reordenação. São representadas por **[ ] ou list( )**. Ex.: cor = **[azul, verde]** → cor.append(“vermelho”) → cor = **[azul, verde, vermelho]**.
* **Dicionários:** Nos dicionários podemos colocar dois valores em cada posição e são criados com **{ }** ou através da classe **dict( ).** São inseridos sempre uma chave e o valor dessa chave. Ex.: frutas = **{“maçã”: “vermelha”, “abacaxi”: “amarela”, “limão”: “verde”}**

## - M**anipulando Arquivos e Pastas**

* **CLI Arguments:** Tem como objetivo ler informações para dentro de um script. Ex.:



* **Criar uma pasta:**

# 

* **Criar pastas e subpastas que ainda não existem:**



* **Acessar uma pasta:**

# 

* **Exibir a pasta atual:**



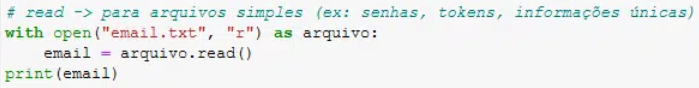
* **Criar um arquivo em branco:**

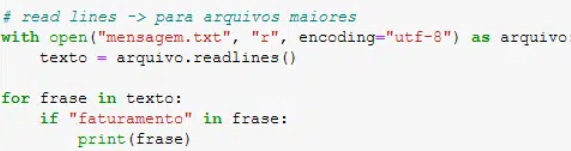


* **Listar arquivos:**

# 

* **Ler arquivos:**





* **Escrever arquivos:**



# 

# LINUX

## - Comandos Básicos

No terminal:

* **Touch**: O comando touch do Linux cria um arquivo vazio. Ex.: touch .arquivo
* **Echo**: O como echo adiciona o texto para um determinado arquivo. Ex.: echo “texto” >> .arquivo

# AMBIENTE VIRTUAL

## - Venv

No terminal:

* **Criando o ambiente virtual**: Dentro da pasta que deseja criar, *python3 -m venv .nomedavenv*(obs.: Ao digitar o nome da venv e inserir “.” antes do nome, ele oculta a pasta).
* **Ativando o ambiente virtual**: A pasta criada da venv tem o arquivo “activate” para ativar o ambiente você deve utilizar o comando *source*. Ex.: source .venv/bin/activate

Fontes:

Hashtagtreinamentos - https://www.hashtagtreinamentos.com/trabalhar-com-arquivos-de-texto-python?gclid=CjwKCAjwrNmWBhA4EiwAHbjEQO20qD3hnyCU1m5CPjPxjblTwSTn\_0QWR30v5-BGzbmML3OKr2310BoC0QQQAvD\_BwE