

código.py

GALÁXIA 9

**Ferramentas
Financeiras**



Introdução

Seja muito bem vindo a Galáxia 9! Nessa Galáxia vamos aprender um pouco sobre Ferramentas Financeiras práticas que você poderá usar no seu dia a dia! No entanto, é importante compreender que há uma variedade infinita de ferramentas financeiras disponíveis. Portanto, o que será apresentado aqui representa apenas uma parte do que considero relevante... Tendo dito isso, vamos começar!



Mundo 1

1. Como vamos construir ferramentas?

Como mencionei na introdução, essa galáxia não tem fim, então sempre que julgar necessário, vou incluir uma nova aula com uma ferramenta adicional aqui.

O enfoque não estará nos exercícios, mas sim na construção de ferramentas práticas e utilizáveis. Meu objetivo é orientá-lo na criação de instrumentos diretamente aplicáveis no mercado financeiro, integrando e aplicando todo o conhecimento adquirido ao longo do curso.

Mundo 2

2. Ferramenta 1: Como pegar dados de balanços das empresas na CVM.

Portal de dados abertos da CVM:

http://dados.cvm.gov.br/dados/CIA_ABERTA/DOC/

Esse site tem muitos dados, vale a pena dar uma olhada. No entanto, nesta aula, vamos nos concentrar exclusivamente nos balanços das empresas, embora o processo seja o mesmo.

Para isso, faremos o download da DFP, que contém vários dados, como BPA, DRE e DFC. Todos esses dados têm versões individuais e consolidadas. A versão individual refere-se à empresa em si, enquanto a consolidada retrata o resultado de todas as empresas que a companhia principal tem mais de 30%. Por exemplo, à Itaúsa, que é uma holding, o consolidado se faz mais que necessário.

A data da DFP corresponde ao fechamento do ano fiscal da empresa e geralmente é divulgada no quarto trimestre, entre outubro, novembro e dezembro. No entanto, algumas empresas têm exceções; é importante verificar essa informação.

É crucial ter um conhecimento mínimo de contabilidade neste momento. Portanto, é essencial compreender que o balanço patrimonial representa uma fotografia daquela situação específica, enquanto as demonstrações, como DRE e DFC, são mais como um filme. Na CVM, os resultados dos trimestres são agregados. Na DRE, apenas o quarto trimestre é somado, sendo necessário subtrair os resultados dos três trimestres anteriores. Já na DFC, a soma ocorre em todos os trimestres, sendo necessário subtrair o resultado do primeiro no segundo, do segundo no terceiro e, por último, do terceiro no quarto trimestre.

Também é relevante mencionar que a CVM geralmente tem um atraso de um ou dois dias.

É possível contornar essa situação, mas para a maioria das pessoas, simplesmente importar a planilha com o código já deve ser suficiente.

```
import pandas as pd
import requests
import os
import zipfile
```

Depois de fazer os imports, faça um range do período desejado e selecione um caminho para que os arquivos sejam salvos:

```
anos = range(2010, 2023)

url_base = f'http://dados.cvm.gov.br/dados/CIA_ABERTA/DOC/'

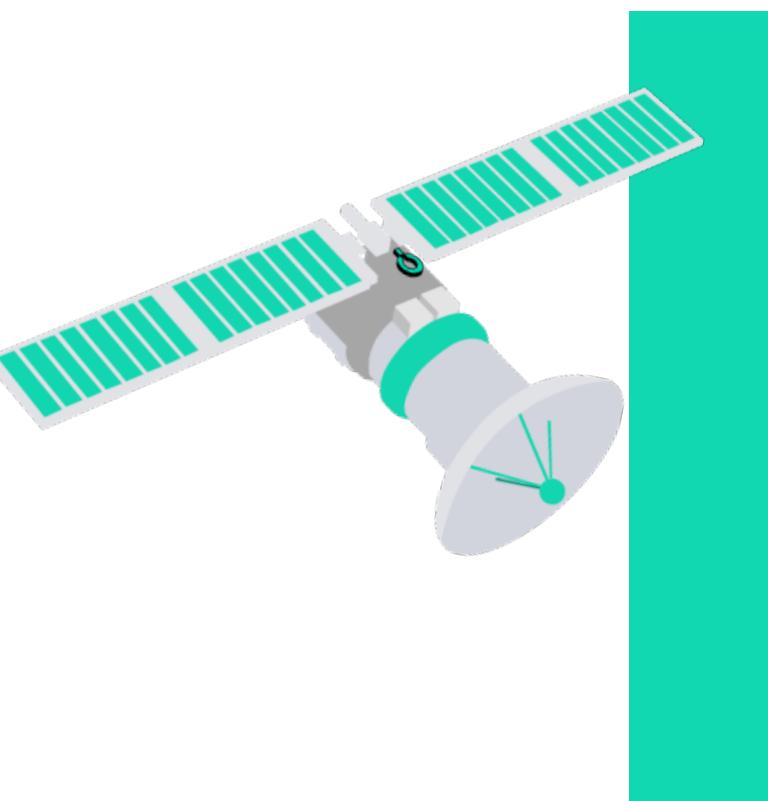
os.chdir(r'C:\Users\lsiqu\dev\github\codigos_curso\Galáxia_9_ferramentas_fin\dados_cvm')
```

Faça o download do Zip com requests.get na URL:

```
for ano in anos:

    download = requests.get(url_base +
        f'DFP/DADOS/dfp_cia_aberta_{ano}.zip')

    open(f"dfp_cia_aberta_{ano}.zip", "wb").write(download.content)
```



Faça uma lista vazia:

```
lista_demonstracoes_2010_2022 = []
```

Então, pegue o diretório atual para encontrar cada zip, “deszipar” e abrir a planilha dentro.

Depois disso, crie uma coluna nova com o tipo do documento, e adicione na base de dados.

```
diretorio_atual = os.getcwd()

for arquivo in os.listdir(diretorio_atual):

    arquivo_zip = zipfile.ZipFile(arquivo)

    tipo = "dfp"

    ano = arquivo[-8: -4]

    for planilha in arquivo_zip.namelist():

        if planilha != f"{tipo}_cia_aberta_{ano}.csv":

            demonstracao = pd.read_csv(arquivo_zip.open(planilha), sep =
                ";", decimal = ",", encoding =
                'ISO-8859-1', chunksize=1000)

            demonstracao = pd.concat(demonstracao, ignore_index=True)

            demonstracao["tipo_doc"] = len(demonstracao) * [tipo]

            lista_demonstracoes_2010_2022.append(demonstracao)
```

código.py

Faça um pd.concat na base de dados.

```
base_dados = pd.concat(lista_demonstracoes_2010_2022)

base_dados
```

Agora separe a coluna que vem com duas informações juntas na CVM, criando duas novas.

```
base_dados[["con_ind", "tipo_dem"]] =
base_dados['GRUPO_DFP'].str.split("-", expand = True)

base_dados["tipo_dem"] = base_dados["tipo_dem"].str.strip()
base_dados["con_ind"] = base_dados["con_ind"].str.strip()
base_dados["con_ind"] = base_dados["con_ind"].astype(str)

base_dados
```

E retire as colunas que não vamos utilizar.

```
base_dados = base_dados.loc[:, ~base_dados.columns.isin(['ST_CONTA_FIXA', 'COLUNA_DF', 'GRUPO_DFP'])]
```

Deixe apenas o último resultado das empresas.

```
base_dados = base_dados[base_dados["ORDEM_EXERC"] != "PENÚLTIMO"]
```

Use esse comando para encontrar o tipo de demonstração.

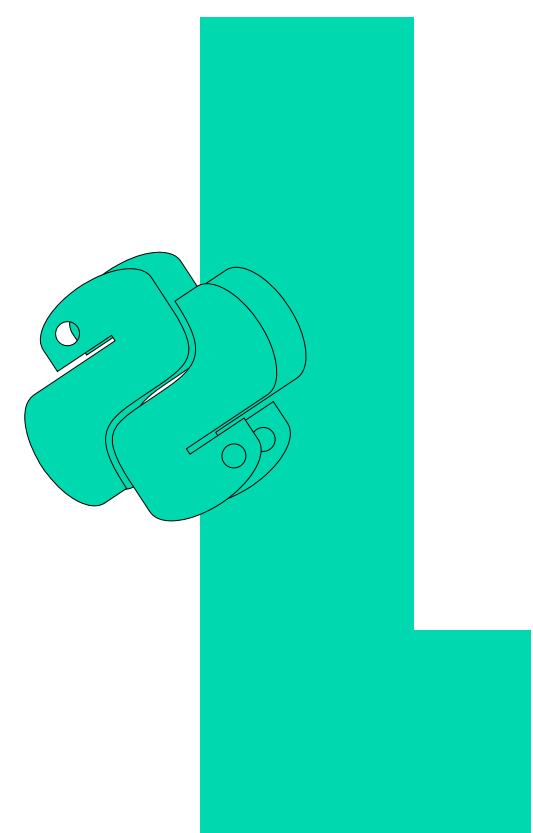
```
#usar o ctrl F
lista_dem = base_dados['tipo_dem'].unique()

lista_dem
```

E esse para encontrar o nome de todas as empresas do período.

```
#usar o ctrl F
lista_empresas = base_dados['DENOM_CIA'].unique()

lista_empresas
```



Por último, use esse código para fazer uma pesquisa específica de uma empresa.

```
weg_dre = base_dados[(base_dados["DENOM_CIA"] == "WEG S.A.") &
                      (base_dados["tipo_dem"] == "Demonstração do Resultado") &
                      (base_dados["tipo_doc"] == "dfp") &
                      (base_dados["DS_CONTA"] == "Receita de Venda de Bens e/ou Serviços") &
                      (base_dados["con_ind"] == "DF Consolidado")]

weg_dre
```

Mundo 3

3. Ferramenta 2: Como obter dados de fundos de investimento na CVM

Nesse mundo vamos seguir a mesma linha e agora obter os dados de fundos de investimento.

Comece falando para o pandas colocar apenas quatro casas decimais.

```
import pandas as pd
import requests
import zipfile
import os

pd.options.display.float_format = '{:.4f}'.format
```

Entre no site da CVM e pegue de exemplo informações do ano que desejar.

```
ano = "2022"
mes = "12"
url = f'https://dados.cvm.gov.br/dados/FI/DOC/INF_DIARIO/DADOS/inf_diario_fi_{ano}{mes}.zip'
```

Faça o download do arquivo.

```
download = requests.get(url)

with open(f"inf_diario_fi_{ano}{mes}.zip", "wb") as arquivo_cvm:
    arquivo_cvm.write(download.content)

arquivo_zip = zipfile.ZipFile(f"inf_diario_fi_{ano}{mes}.zip")
```



Use o read.csv para abrir a planilha.

```
dados_fundos = pd.read_csv(arquivo_zip.open(arquivo_zip.namelist()[0]),
sep = ";", encoding = 'ISO-8859-1')

dados_fundos
```

Agora, faça o download das informações de cadastro para depois associar o CNPJ da empresa com o seu nome.

```
dados_cadastro =
pd.read_csv('https://dados.cvm.gov.br/dados/FI/CAD/DADOS/cad_fi.csv',
sep = ";", encoding = 'ISO-8859-1')

dados_cadastro = dados_cadastro[['CNPJ_FUNDO', 'DENOM_SOCIAL']]

dados_cadastro = dados_cadastro.drop_duplicates()

dados_cadastro
```

É possível também pegar apenas a cota de início e fim do mês.

```
data_inicio_mes = (dados_fundos['DT_COMPTC'].sort_values(ascending = True).unique())[-0]
data_fim_mes = (dados_fundos['DT_COMPTC'].sort_values(ascending = True).unique())[-1]

dados_fundos_filtrado =
dados_fundos[(dados_fundos['DT_COMPTC'].isin([data_inicio_mes,
data_fim_mes]))]

dados_fundos_filtrado
```

Agora junte os dois dataframes para associar o CNPJ da empresa com o seu nome.

```
base_final = pd.merge(dados_fundos_filtrado, dados_cadastro, how =
"left",
left_on = ["CNPJ_FUNDO"], right_on =
["CNPJ_FUNDO"])

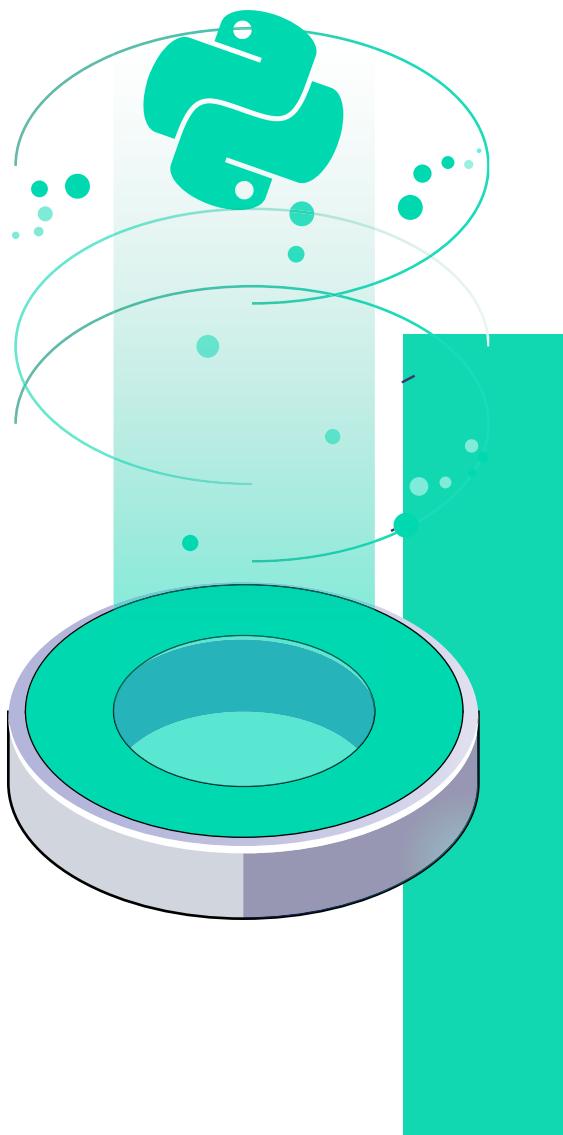
base_final = base_final[['CNPJ_FUNDO', 'DENOM_SOCIAL',
'DT_COMPTC', 'VL_QUOTA', 'VL_PATRIM_LIQ',
'NR_COTST']]

base_final = base_final.dropna()

base_final
```

Por último, caso queira, também é possível buscar um fundo específico pelo nome.

```
base_final[base_final['DENOM_SOCIAL'].str.contains("ARCA GRÃO")]
```



Mundo 4

4. Ferramenta 3: Como calcular o beta das empresas? (Regressão linear no Python)

O beta é uma medida que expressa a relação entre o desempenho de uma ação e o mercado em geral. É a representação estatística do retorno de uma ação em comparação com o retorno do mercado, geralmente medido por um índice renomado como o S&P 500 ou o IBOV.

Por definição, o beta do mercado é 1, e uma ação pode ter um beta maior ou menor do que 1. Por exemplo, uma ação com beta 2, se o mercado sobe 10%, essa ação tende a subir 20%. Se o beta for 0,5, ela provavelmente subirá 5% com um aumento de 10% no mercado. O beta é uma métrica de risco, onde risco é fundamentalmente associado à volatilidade.

Por outro lado, a regressão linear no Python é uma abordagem para tentar explicar uma variável considerando as variações de outra. Geralmente, é usada para compreender o retorno de uma ação com base no desempenho do mercado.

No entanto, é crucial ter cautela, pois a correlação não implica em causalidade. Você pode encontrar um modelo que pareça muito preciso, mas pode ser apenas uma coincidência. Por exemplo, se toda vez que chove em São Paulo a bolsa sobe, isso não tem necessariamente uma base lógica.

Para começar use um pacote novo, o statsmodels, para que consiga fazer uma regressão linear.

```
import yfinance as yf
import statsmodels.api as sm
from datetime import timedelta
from datetime import datetime
```

Pegue uma ação, um índice como mercado e um período de tempo em dias e faça o download.

```
ativos = ["WEGE3.SA", "^BVSP"]
data = datetime.now()
tres_anos_atras = data - timedelta(days = 1095)

dados_cotacoes = yf.download(ativos, tres_anos_atras, data)['Adj Close']

dados_cotacoes
```

Realize os retornos diários.

```
retornos_diarios = dados_cotacoes.pct_change().dropna()

retornos_diarios
```

Agora, crie uma regressão linear. X como retorno do mercado e Y como retorno do ativo. Sendo **Y = Alfa + Beta * X**, com Alfa sendo uma constante

```
X = retornos_diarios['^BVSP']
Y = retornos_diarios[ativos[0]]
X = sm.add_constant(X)
model = sm.OLS(Y, X).fit()
```

E aqui está o resultado de tudo o que você precisa saber sobre regressões lineares.

```
# Beta
print(model.params[1])
print(model.rsquared)
print("-----")
print(str(model.summary()))
```

Mundo 5

5. Ferramenta 4: Como fazer uma otimização de Markowitz (otimização de carteiras no Python)

A contribuição de Markowitz ao relacionar risco e retorno foi pioneira, demonstrando que ao aumentar o risco em uma carteira, a expectativa de retorno também cresce, algo que hoje é uma noção comum. A otimização de Markowitz procura equilibrar essa relação.

No entanto, essa otimização tem algumas falhas, como por exemplo, utilizar o retorno e a volatilidade histórica para projetar o futuro. Isso faz com que em períodos de volatilidades muito altas ou baixas, a otimização diminua o peso desse ativo em projeções futuras. Porém é importante saber formalizar essa questão de risco e retorno dentro do Python e é interessante conseguir visualizar isso em gráficos e números.

Outro aspecto é o índice Sharpe, que avalia a rentabilidade em relação à volatilidade. Por exemplo, se um ativo teve um retorno de 20% em 6 meses com volatilidade de 10%, o Sharpe seria 2. Isso é relevante, já que na otimização de Markowitz, busca-se a carteira com o maior Sharpe, pois quanto maior o retorno e menor a volatilidade, melhor.

Outro novo pacote que é necessário utilizar é o `spicy.optimize`, que busca otimizar e criar uma Fronteira Eficiente. Uma ideia que mostra as melhores maneiras de combinar diferentes investimentos para criar uma carteira inteligente. Ela foca nas combinações que oferecem o melhor retorno para o menor risco possível.

```
import yfinance as yf
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import pandas as pd
from scipy.optimize import minimize
import matplotlib.ticker as mtick
```

Selecione o período de datas de início e fim. Quanto maior o período, mais coisas aleatórias virão no seu histórico, mas também ficará mais robusto.

```
inicio = dt.date(2015, 1, 1)
final = dt.date(2022, 12, 31)
```

Puxe as ações que desejar.

Obs.: O Mercado Americano é melhor para fazer a otimização pois é mais consolidado e normal. Então aqui escolhemos 4 ações de lá.

```
lista_acoes = ["AAPL", "NKE", "GOOGL", "AMZN"]
precos = yf.download(lista_acoes, inicio, final)['Adj Close']
precos
```

Aqui é necessário que faça o retorno logarítmico, por possuir uma vantagem em cima do retorno aritmético. Em Log pode somar os retornos diários, coisa que não é possível fazer no aritmético.

Depois, faça a média dos retornos de cada ação e sua matriz co-variância, para calcular a projeção de retornos e de volatilidade, respectivamente.

```
retornos = precos.pct_change().apply(lambda x: np.log(1+x)).dropna()
media_retornos = retornos.mean()
matriz_cov = retornos.cov()

media_retornos
```

Crie uma simulação. Quanto maior o número de carteiras, melhor. Então “chute” aleatoriamente 100.000 pesos diferentes para encontrar o melhor.

Calcule o retorno anual multiplicando por 252 dias úteis. Pesos só podem ser entre 0 e 1. No final, retorne o sharpe para cada carteira gerada.

```
numero_carteiras = 100000
vetor_retornos_esperados = np.zeros(numero_carteiras)
vetor_volatilidades_esperadas = np.zeros(numero_carteiras)
vetor_sharpe = np.zeros(numero_carteiras)
tabela_pesos = np.zeros((numero_carteiras, len(lista_acoes)))

for k in range(numero_carteiras):

    pesos = np.random.random(len(lista_acoes))
    pesos = pesos/np.sum(pesos) #só pode ter 100%
    tabela_pesos[k, :] = pesos

    vetor_retornos_esperados[k] = np.sum(media_retornos * pesos * 252)
    vetor_volatilidades_esperadas[k] = np.sqrt(np.dot(pesos.T,
np.dot(matriz_cov*252, pesos)))

    vetor_sharpe[k] =
    vetor_retornos_esperados[k]/vetor_volatilidades_esperadas[k]
    vetor_sharpe
```

Descubra os pesos da melhor carteira segundo Markowitz.

```
indice_do_sharpe_maximo = vetor_sharpe.argmax()
tabela_pesos[indice_do_sharpe_maximo]
```

Tire do logaritmo e transforme em exponencial para descobrir o retorno esperado ao ano da carteira.

```
tabela_retornos_esperados_arit = np.exp(vetor_retornos_esperados) - 1

tabela_retornos_esperados_arit
```

Faça a Fronteira Eficiente.

```

eixo_y_fronteira_eficiente =
np.linspace(tabela_retornos Esperados_arit.min(),
tabela_retornos Esperados_arit.max(), 50)

def pegando_retorno(peso_teste):
    peso_teste = np.array(peso_teste)
    retorno = np.sum(media_retornos * peso_teste) * 252
    retorno = np.exp(retorno) - 1
    return retorno

def checando_soma_pesos(peso_teste):
    return np.sum(peso_teste)-1

def pegando_vol(peso_teste):
    peso_teste = np.array(peso_teste)
    vol = np.sqrt(np.dot(peso_teste.T, np.dot(matriz_cov*252,
peso_teste)))
    return vol

peso_inicial = [1/len(lista_acoes)] * len(lista_acoes)

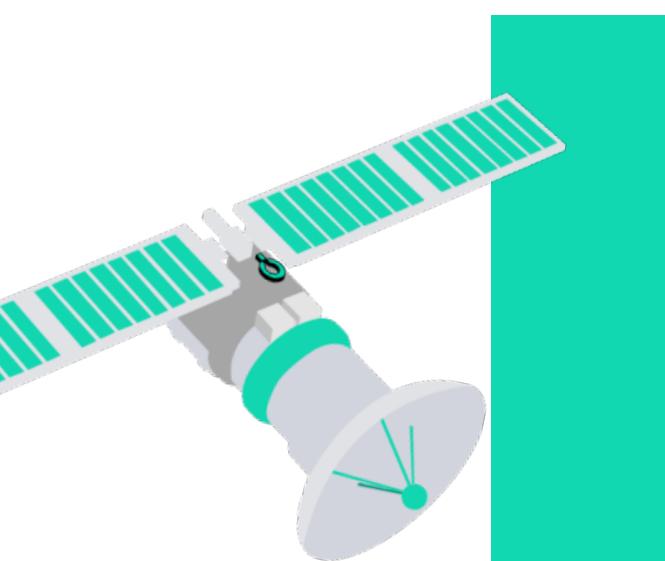
limites = tuple([(0, 1) for ativo in lista_acoes])

eixo_x_fronteira_eficiente = []

for retorno_possivel in eixo_y_fronteira_eficiente:
    #vamos pegar a melhor volatilidade para cada retorno possivel

    restricoes = ({'type':'eq', 'fun':checando_soma_pesos},
                  {'type':'eq', 'fun': lambda w: pegando_retorno(w) -
retorno_possivel})
    result = minimize(pegando_vol, peso_inicial, method='SLSQP',
bounds=limites,
                      constraints = restricoes)
    eixo_x_fronteira_eficiente.append(result['fun'])


```



A seguir, faça o plot do gráfico.

```

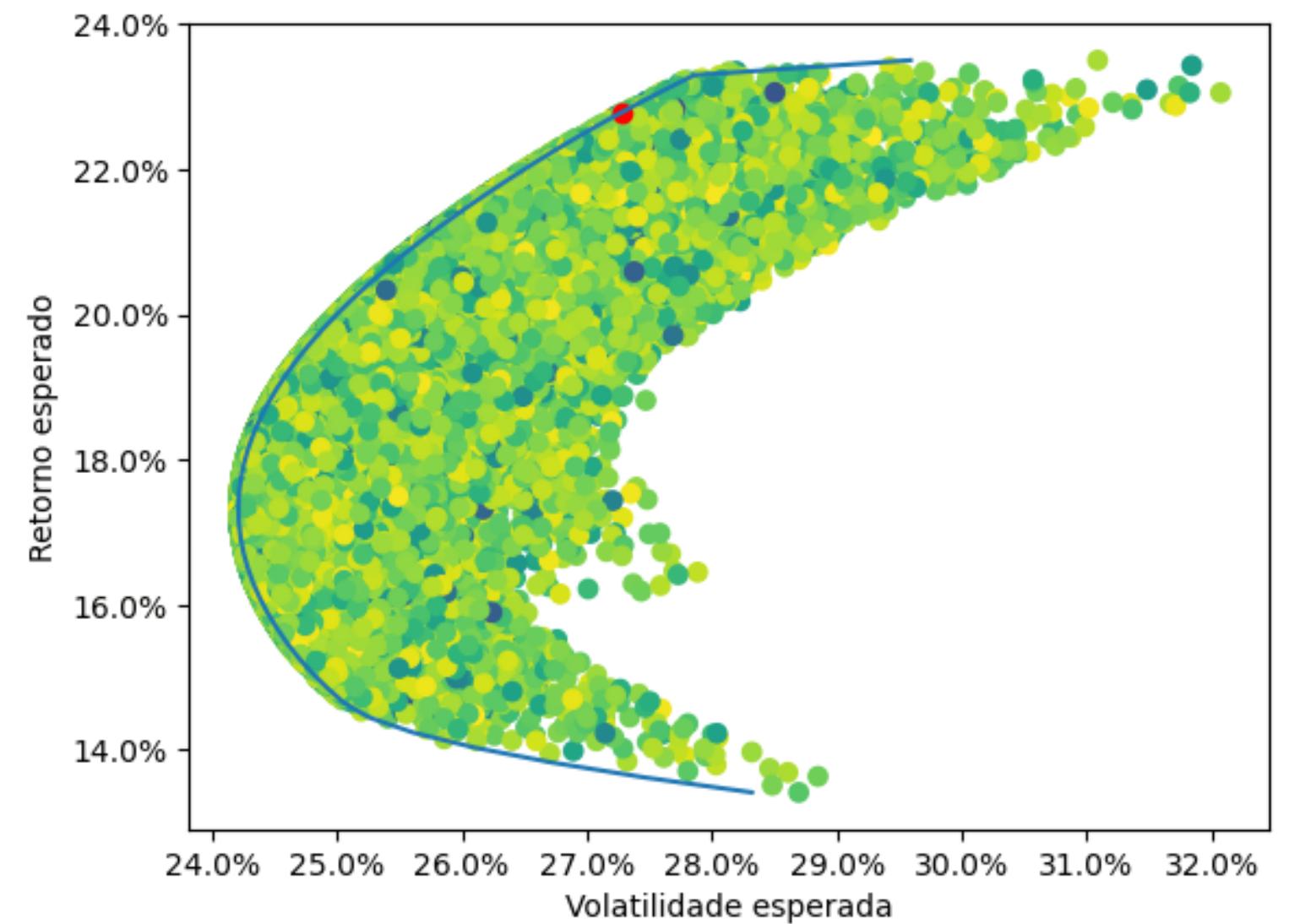
fig, ax = plt.subplots()

ax.scatter(vetor_volatilidades Esperadas,
tabela_retornos Esperados_arit, c = tabela_sharpe)
plt.xlabel("Volatilidade esperada")
plt.ylabel("Retorno esperado")
ax.scatter(vetor_volatilidades Esperadas[indice_do_sharpe_maximo],
tabela_retornos Esperados_arit[indice_do_sharpe_maximo], c
= "red")
ax.plot(eixo_x_fronteira_eficiente, eixo_y_fronteira_eficiente)
ax.yaxis.set_major_formatter(mtick.PercentFormatter(1.0))
ax.xaxis.set_major_formatter(mtick.PercentFormatter(1.0))

plt.show()

```

O ponto vermelho representa a melhor carteira, a carteira eficiente.



Mundo 6

6. Ferramenta 5: Simulação de Monte Carlo – Calculando VAR e probabilidade lucro de um investimento

Nada mais é que pegar um grande número de simulações e conseguir calcular as estatísticas, como por exemplo, descobrir em uma carteira de 10 ações, quanto é o máximo que se pode perder no próximo ano com uma confiança de 95% e qual a probabilidade de obter lucro.

Lembrando que sempre existe a chance de dar errado, nunca vai ter um intervalo de 100% de confiança, que são os eventos extremos, porém sempre tente se prevenir e se preparar.

```
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import yfinance as yf
from numpy import linalg as LA
```

Para isso, após importar os pacotes necessários pegue 5 ações no período de 300 dias, para ter o preço ajustado.

```
#pegando dados

lista_acoes = ['WEGE3', 'PCAR3', 'LREN3', 'PETR4', 'VALE3']
lista_acoes = [acao + ".SA" for acao in lista_acoes]

data_final = dt.datetime.now()
data_inicial = data_final - dt.timedelta(days=300)

precos = yf.download(lista_acoes, data_inicial, data_final)['Adj Close']
```

Calcule retornos pegando matriz de covariância, dessa vez sem usar o retorno logaritmo, e no final retorne os pesos das carteiras.

```
retornos = precos.pct_change().dropna()
media_retornos = retornos.mean()
matriz_covariancia = retornos.cov()
pesos_carteira = np.full(len(lista_acoes), 1/len(lista_acoes))
numero_acoes = len(lista_acoes)

pesos_carteira
```

Para criar as simulações precisa de retornos futuros sintéticos, com isso temos a seguinte fórmula:

$$\text{Retornos_sintéticos} = \text{média_retornos} + \text{Rpdf} \times \text{L}.$$

* média_retornos = Média dos retornos.

* Rpdf = Matriz aleatória gerada por alguma função de densidade de probabilidade.

* L = Matriz triangular inferior proveniente de uma decomposição de Cholesky, usando como base a matriz de covariância dos dados originais.

Por que fazer isso?

Nós assumimos que a distribuição de retornos é uma distribuição normal multivariada (isso é uma expansão da distribuição normal).

Quando geramos retornos aleatórios de cada ativo (Weg, Vale, etc), nós criamos vetores aleatórios descorrelacionados entre si. Para corrigir isso, precisamos correlacionar esses retornos (no mundo real isso é o que acontece) e, para isso, usamos a matriz triangular obtida a partir da covariância.

O que vai acontecer com 1000 reais nos próximos 252 dias com 10.000 simulações?

```
#premissas montecarlo
numero_simulacoes = 10000
dias_projetados = 252
capital_inicial = 1000
```

Calcule o retorno médio.

```
retorno_medio = retornos.mean(axis = 0).to_numpy()
matriz_retorno_medio = retorno_medio * np.ones(shape =
(dias_projetados, numero_acoes))
```

Matriz de covariância da fórmula.

```
#gerando L
L = LA.cholesky(matriz_covariancia)
L
```

Faça as simulações.

```
retornos_carteira = np.zeros([dias_projetados, numero_simulacoes])
#cada coluna é uma simulação
montante_final = np.zeros(numero_simulacoes)

for s in range(numero_simulacoes):
    Rpdf = np.random.normal(size=(dias_projetados, numero_acoes))

    retornos_sinteticos = matriz_retorno_medio + np.inner(Rpdf, L)
    #única coisa random é o Rpdf

    retornos_carteira[:, s] = np.cumprod(np.inner(pesos_carteira,
                                                   retornos_sinteticos)
                                         + 1) * capital_inicial
    montante_final[s] = retornos_carteira[-1, s]

retornos_carteira
```

Faça o plot dos gráficos.

```
plt.plot(retornos_carteira, linewidth=1)
plt.ylabel('Dinheiro')
plt.xlabel('Dias')
plt.show()
```



E por último, calcule algumas probabilidades.

```
montante_99 = str(np.percentile(montante_final, 1))
montante_95 = str(np.percentile(montante_final, 5))
montante_mediano = str(np.percentile(montante_final, 50))
cenarios_com_lucro = str((len(montante_final[montante_final > 1000])/
                           len(montante_final)) * 100) + "%"
```

Monte um print para devolver os cálculos das estatísticas.

```
print(f'''Ao investir R$ 1000,00 na carteira {lista_acoes},
podemos esperar esses resultados para os próximo ano,
utilizando o método de Monte Carlo com 10 mil simulações:

Com 50% de probabilidade, o montante será maior que R$ {montante_mediano}.

Com 95% de probabilidade, o montante será maior que R$ {montante_95}.

Com 99% de probabilidade, o montante será maior que R$ {montante_99}.

Em {cenarios_com_lucro} dos cenários, foi possível obter lucro no
próximo ano.'''')
```

Utilidades

A partir dessas métricas, é possível calcular coisas como o VAR da carteira para diferentes intervalos de confiança e medir se o risco que você está correndo é compatível com o que você aguenta perder. No caso dessa carteira, uma queda de 27,5% está dentro dos 95% dos cenários mais possíveis de acontecer no próximo ano.

Faça a distribuição de montantes finais com as simulações de Monte Carlo.

```
config = dict(histtype = "stepfilled", alpha = 0.8, density = False,
              bins = 150)
fig, ax = plt.subplots()
ax.hist(montante_final, **config)
ax.xaxis.set_major_formatter('R${x:.0f}')
plt.title('Distribuição montantes finais com simulação MC')
plt.xlabel('Montante final (R$)')
plt.ylabel("Frequência")
plt.show()
```

Mundo 7

7. Ferramenta 6: Como calcular Max Drawdown no Python

Calcular o máximo Drawdown é simples mas super útil para o seu dia a dia. Vamos aprender a descobrir quanto a pessoa mais azarada teria perdido em uma ação ou fundo se comprasse na máxima histórica e vendesse na baixa. Quanto menor o drawdown, claro que é melhor.

Importe novamente alguns dos mesmos pacotes de sempre.

```
import yfinance as yf
import matplotlib.pyplot as plt
import datetime as dt
import matplotlib.ticker as mtick
import matplotlib.dates as mdate
import mplcyberpunk
```

Puxe 1500 dias de cotação de PETR4, ou qualquer ação que desejar.

```
data_final = dt.datetime.now()
data_inicial = data_final - dt.timedelta(days=1500)
ativo = "PETR4.SA"
precos = yf.download(ativo, data_inicial, data_final)['Adj Close']
```

Aqui pegue o preço máximo acumulativo. Calcule o preço atual, diminua da máxima histórica até aquele momento e divida pela máxima histórica até aquele momento.

Descubra o drawdown máximo, que seria a pessoa que mais perdeu dinheiro nessa ação.

```
precos_max = precos.cummax()
drawdowns = (precos - precos_max)/precos_max
drawdown_maximo = drawdowns.min()
print(drawdown_maximo)
```

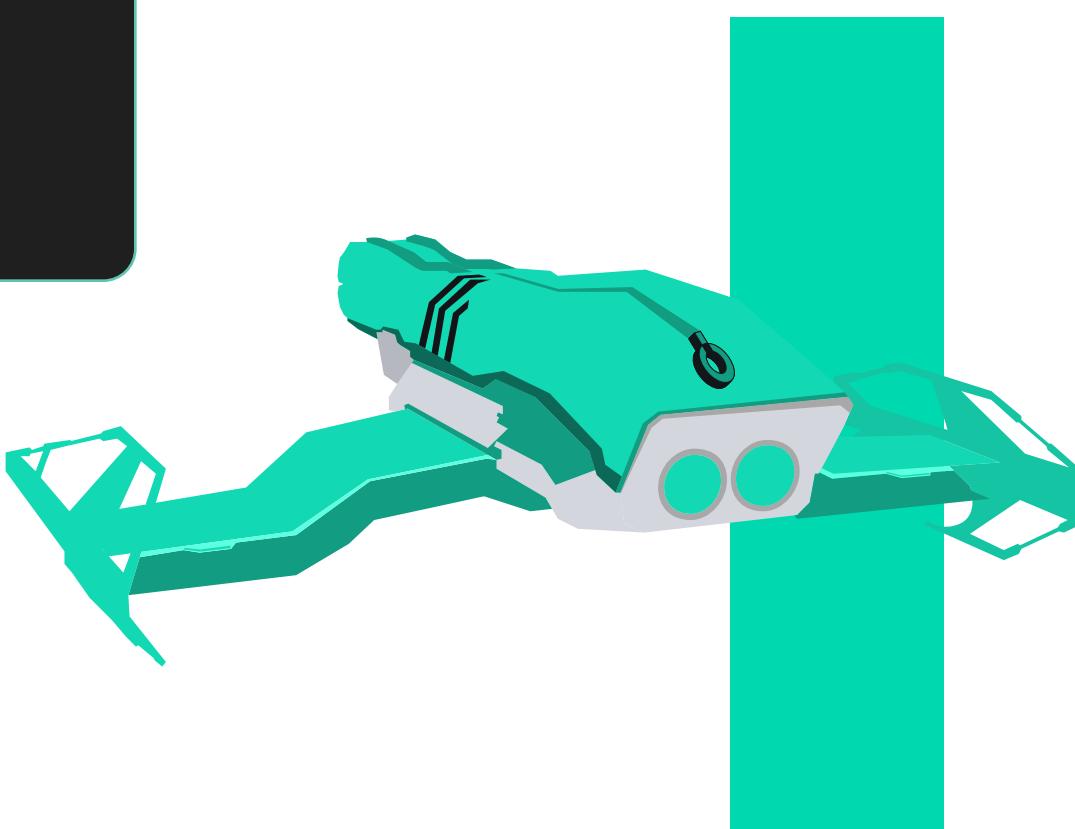
Se o Drawdown for 0 o preço está na máxima histórica.

```
drawdowns.head(50)
```

Também é possível fazer o gráfico de underwater, na qual toda vez que a linha “toca” o eixo 0%, ninguém está perdendo dinheiro na ação, pois está na máxima histórica, e o resto seria quanto a pessoa perderia se tivesse investido nessa mesma máxima.

```
plt.style.use("cyberpunk")

fig, ax = plt.subplots()
ax.plot(drawdowns.index, drawdowns)
ax.yaxis.set_major_formatter(mtick.PercentFormatter(1.0))
ax.xaxis.set_major_locator(mdate.YearLocator(1))
plt.show()
```



Mundo 8

8. Ferramenta 7: Calculando o custo de capital (CAPM) de criptomoedas

Como calcular o CAPM:

Retorno esperado = Renda Fixa + Beta (Prêmio de risco - Renda Fixa)

Renda fixa, vai ser o Treasury Bond de 10 anos dos Estados Unidos.

Beta = 1, por o Bitcoin ser o benchmark das criptomoedas.

Prêmio de risco vamos calcular.

Primeiro instale e utilize o pacote cryptocmd para que consiga pegar os dados de criptomoedas, inclusive até mesmo as mais diferentes que existem.

```
pip install cryptocmd
```

E importe os pacotes que já conhecemos.

```
import pandas as pd
import numpy as np
import yfinance as yf
from cryptocmd import CmcScraper
import statsmodels.api as sm
```

Faça um scraper para pegar os dados do Bitcoin e depois transformar o índice na data.

```
scraper = CmcScraper("BTC")
df = scraper.get_dataframe()
df = df.set_index("Date")
```

df

Pegue os dados da renda fixa (bond) dos Estados Unidos de 10 anos desde 2017, para calcular o prêmio de risco.

```
tbond10 = yf.download('IEF', "2017-01-01", "2023-04-23")['Adj Close']
tbond10 = tbond10.resample("Y").last().pct_change().dropna()
tbond10
```

E os dados do fechamento anual do Bitcoin desde 2017.

```
df_bitcoin_anual = df.resample("Y").last().pct_change().dropna()
df_bitcoin_anual = df_bitcoin_anual['Close']
df_bitcoin_anual = df_bitcoin_anual.loc[df_bitcoin_anual.index > "2017-12-31"]
df_bitcoin_anual
```

Calcule o retorno anual do Bitcoin menos o retorno anual da renda fixa Americana e faça a média.

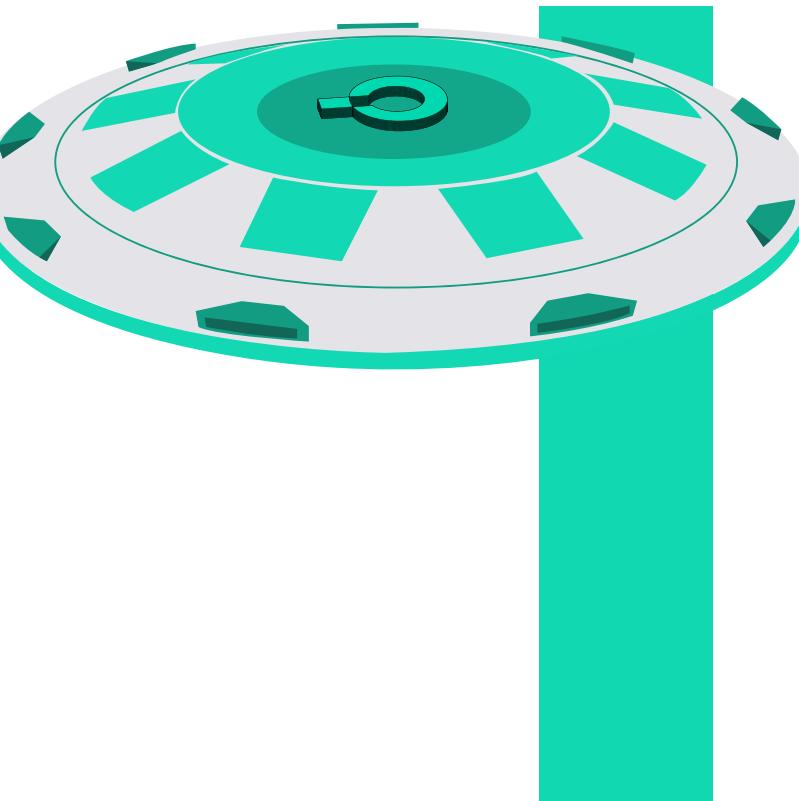
```
premio_risco = df_bitcoin_anual - tbond10  
premio_risco.mean()
```

CAPM vai ser o T-bond mais o beta vezes a média do prêmio de risco menos a renda fixa.

```
retorno_esperado = tbond10.loc['2023-12-31'] + 1 * (premio_risco.mean()  
- tbond10.loc['2023-12-31'])  
  
retorno_esperado
```

Calcule agora o Beta do Ethereum:

```
scraper = CmcScraper("ETH")  
df_eth = scraper.get_dataframe()  
df_eth = df_eth.set_index("Date")  
  
df_eth
```



Compare com o Bitcoin.

```
df_btc = df.loc[df.index >= "2015-08-07"]  
df_eth = df_eth[df_eth.index >= "2015-08-07"]
```

Retornos das duas criptomoedas.

```
retornos_eth = df_eth['Close'].pct_change().dropna()  
retornos_btc = df_btc['Close'].pct_change().dropna()
```

Calcule os betas.

```
X = retornos_btc  
Y = retornos_eth  
X = sm.add_constant(X)  
model = sm.OLS(Y, X).fit()  
  
print(model.params[1])  
print(model.rsquared)
```

Mundo 9

9. Ferramenta 8: Como pegar cotações históricas na B3 de QUALQUER ativo financeiro negociado pela bolsa (ações, opções, etc)

As séries históricas estão disponíveis em:

http://www.b3.com.br/pt_br/market-data-e-indices/servicos-de-dados/market-data/historico/mercado-a-vista/series-historicas/

A parte de baixar as cotações infelizmente não é possível automatizar com Python por conta das limitações do site, então primeiro entre no site e baixe o ano que desejar.

Importe apenas pandas.

```
import pandas as pd
```

Coloque o nome do arquivo baixado.

```
arquivo_bovespa = 'COTAHIST_A2022.TXT'
```

Estrutura para tratar o arquivo disponível em:

http://www.b3.com.br/data/files/33/67/B9/50/D84057102C784E-47AC094EA8/SeriesHistoricas_Layout.pdf

Para facilitar, aqui já estão as informações dos tamanhos dos campos.

```
tamanho_campos = [2,8,2,12,3,12,10,3,4,13,13,13,13,13,13,13,5,18,18,13,1,8,7,13,12,3]

dados_acoes = pd.read_fwf(arquivo_bovespa, widths=tamanho_campos, header=0)
```

Retorne os dados_acoes, com um dataframe com as informações do ano escolhido.

```
dados_acoes
```

Mas, ainda não está legal, então trate este Df. Renomeie as colunas, eliminando a última linha que não é necessária e ajuste os valores com vírgula.

```
## Nomear as colunas

dados_acoes.columns = [
    "tipo_registro",
    "data_pregao",
    "cod_bdi",
    "cod_negociacao",
    "tipo_mercado",
    "noma_empresa",
    "especificacao_papel",
    "prazo_dias_merc_termo",
    "moeda_referencia",
    "preco_abertura",
    "preco_maximo",
    "preco_minimo",
    "preco_medio",
    "preco_ultimo_negocio",
    "preco_melhor_oferta_compra",
    "preco_melhor_oferta_venda",
    "numero_negocios",
    "quantidade_papeis_negociados",
    "volume_total_negociado",
    "preco_exercicio",
    "indicador_correcacao_precos",
    "data_vencimento",
    "fator_cotacao",
    "preco_exercicioPontos",
    "codigo_isin",
    "num_distribuicao_papel"]]

# Eliminar a última linha
linha=len(dados_acoes["data_pregao"])
dados_acoes=dados_acoes.drop(linha-1)

# Ajustar valores com vírgula (dividir os valores dessas colunas por 100)
listaVirgula=[ 
    "preco_abertura",
    "preco_maximo",
    "preco_minimo",
    "preco_medio",
    "preco_ultimo_negocio",
    "preco_melhor_oferta_compra",
    "preco_melhor_oferta_venda",
    "volume_total_negociado",
    "preco_exercicio",
    "preco_exercicioPontos"
]

for coluna in listaVirgula:
    dados_acoes[coluna]=[i/100. for i in dados_acoes[coluna]]
```

Retorne os dados das ações, depois do tratamento.

```
dados_acoes.head(50)
```

Informação importante: Tipos de mercado

- * 10 - Mercado à vista
- * 70 - Opções de Compra
- * 80 - Opções de Venda

Com isso você pode filtrar somente as opções de compra, por exemplo.

```
dados_acoes[dados_acoes['tipo_mercado']==70]
```

Pode filtrar por ações também.

```
WEGE3 = dados_acoes[(dados_acoes['tipo_mercado']==10)&  
| | | | | (dados_acoes['cod_negociacao']=='WEGE3')]
```

E por data de pregão.

```
WEGE3 = WEGE3.set_index('data_pregao')
```

WEGE3

Para deixar melhor ainda, pode transformar a data em datetime.

```
WEGE3.index = pd.to_datetime(WEGE3.index, format = "%Y%m%d")
```

WEGE3

Por último, pode filtrar por mês.

```
WEGE3[WEGE3.index.month == 5]
```

Detalhe importante, é que as cotações não são ajustadas, então caso queira fazer uma base com isso, é necessário pegar os eventos societários como dividendos e splits da empresa e corrigir esses preços.

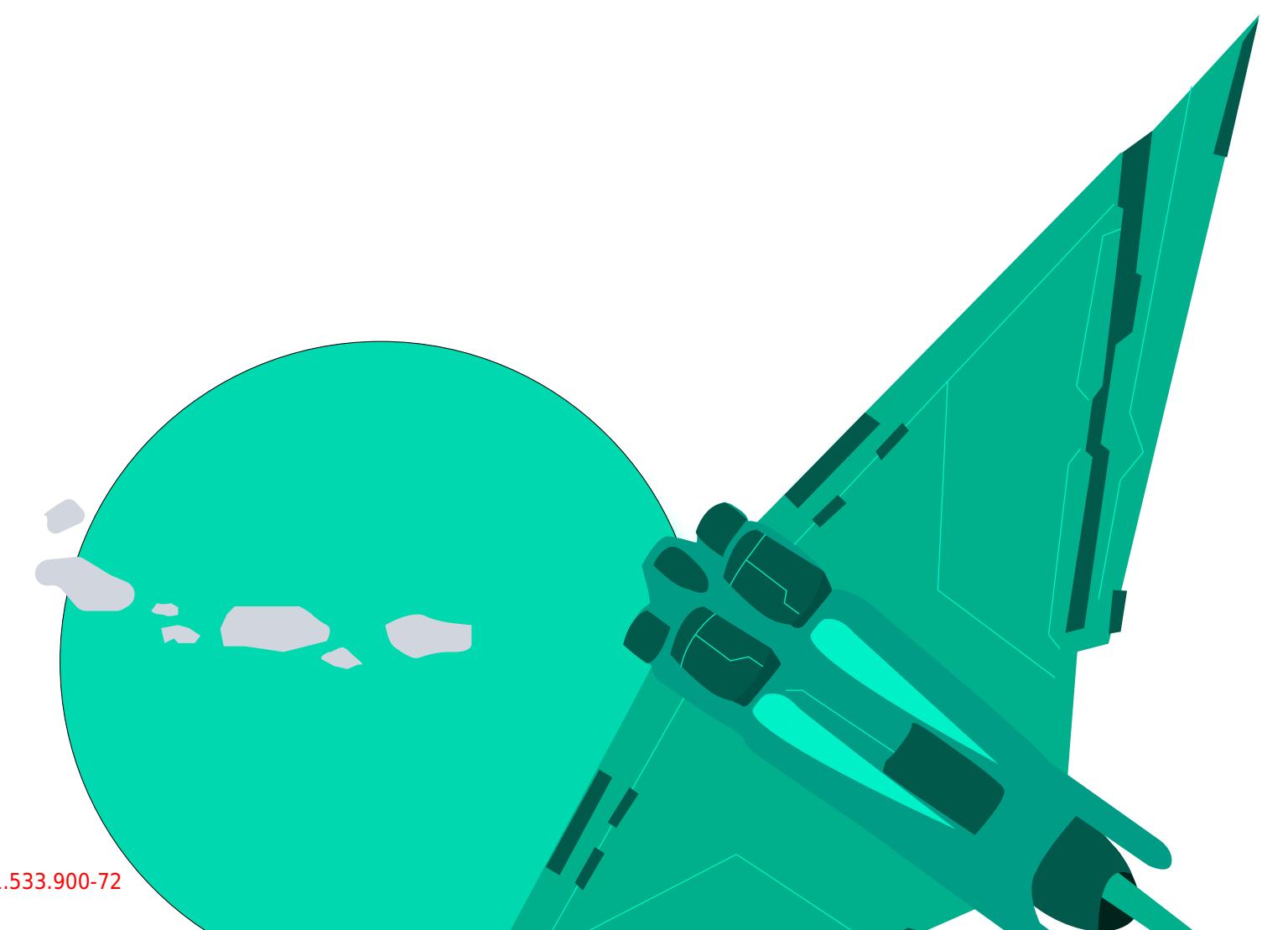
Mundo 10

10. Ferramenta 9: Como puxar o Risco Brasil (CDS)

CDS é o Risco Brasil, basicamente seria o que os estrangeiros estão achando do nosso País, se estão achando muito arriscado investir aqui, qual o prêmio de risco e quanto o Brasil deveria pagar a mais pelo risco político e econômico, entre outras.

Importe os pacotes para fazer Web Scraping.

```
from urllib.request import Request, urlopen  
from bs4 import BeautifulSoup  
import pandas as pd
```



Importe na lista o cds de vários períodos.

```
lista_cds = ['cds-1-year', 'cds-2-years', 'cds-3-years',  
             | | | | 'cds-4-years', 'cds-5-years', 'cds-7-years', 'cds-10-years']
```

Coloque headers pois às vezes o site reconhece que é um bot pegando os dados, e com o header é possível camuflar melhor isso.

```
headers = {'user-agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36'}
```

Abra o navegador e pegue com o request todos os cds que colocou na lista. O código deve conseguir encontrar a tabela dentro do site e pegar as colunas “último” e “data”, transformando a data em índice e depois em datetime e renomear as colunas.

```
lista_dfs = []  
  
for ano_cds in lista_cds:  
  
    url =  
    f'https://br.investing.com/rates-bonds/brazil-{ano_cds}-usd-historical-  
    data'  
  
    req = Request(url, headers=headers)  
    page = urlopen(req)  
    #isso aqui é uma forma de abrir o navegador como se fosse o  
    selenium mas mais otimizado  
    soup = BeautifulSoup(page, features='lxml')  
    table = soup.find_all("table")[1]  
  
    df_cds = pd.read_html(str(table))[0][['Último', 'Data']]  
    df_cds = df_cds.set_index("Data")  
    df_cds.index = pd.to_datetime(df_cds.index, format =  
    "%d.%m.%Y")  
  
    df_cds.columns = [ano_cds]  
  
    lista_dfs.append(df_cds)  
  
base_cds = pd.concat(lista_dfs, axis = 1)  
  
base_cds
```

Isso é uma ótima proxy de risco-país, podendo ser utilizado para calcular risco de capital, fator de risco em factor investing, entre outros.

Mundo 11

11. Como enviar e-mail com o Python: envio automático

A primeira coisa a se fazer é configurar o email, podendo ser qualquer email. O que é necessário entender é que você está conectando o seu servidor ao backend do email que vai enviar. O seu servidor nada mais é que seu computador, então precisa conectar o seu computador com o servidor que você deseja enviar o email.

A conta do Google tem que ter a verificação de 2 fatores ativada para conseguir enviar email com python:

<https://myaccount.google.com/apppasswords>

Isso vai gerar uma senha diferente da sua do Gmail, essa será a senha que você pode colocar o .env, que já foi explicado anteriormente aqui no curso, como fazer.

A biblioteca smtplib permitirá se conectar com o Gmail.

```
import os
import smtplib
from email.message import EmailMessage
from dotenv import load_dotenv
```

Execute essa linha para permitir trazer a senha do .env para esse programa, como já foi ensinado anteriormente.

```
load_dotenv()
```

Coloque o seu email aqui. Também funciona com email institucional.

```
email = "seu_email@gmail.com"
```

Puxe a senha.

```
senha_email = os.environ.get("senha_email")
```

Coloque os destinatários.

```
emails = ['brenno@varos.com.br', "outroemail@hotmail.com"]
```

Escreva informações importantes como assunto, remetente, destinatário e mensagem. Obs.: A mensagem pode ser enviada no formato html também.

```
msg = EmailMessage()
msg['Subject'] = 'Enviando e-mail com Python'
msg['From'] = 'brenno@varos.com.br'
msg['To'] = ", ".join(emails)
msg.set_content('''Segue o relatório diário'''')
```

Anexe imagens.

```
with open('dolar.png', 'rb') as content_file:
    content = content_file.read()
    msg.add_attachment(content, maintype='application', subtype='png', filename='dolar.png')
```

Anexe um PDF agora. Sendo as únicas mudanças o nome do arquivo e seu subtipo.

```
with open('relatorio_diario.pdf', 'rb') as content_file:  
    content = content_file.read()  
    msg.add_attachment(content, maintype='application', subtype='pdf',  
    filename='relatorio_diario.pdf')
```

Aqui caso esteja fazendo com outlook, hotmail ou qualquer outro email, pesquise qual o código, servidor e como enviar email no Google e altere, é muito simples.

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:  
  
    smtp.login(email, senha_do_email)  
    smtp.send_message(msg)
```

Mundo 12

12. Como criar um PDF 100% com Python: Criando um relatório de gestão na prática

Gerar PDF com o Python pode não ser tão simples, mas a integração que ele permite fazer é excelente.

Regulamentação FPDF:

<https://pyfpdf.readthedocs.io/en/latest/reference/FPDF/index.html>

O PDF dentro do Python é um objeto. O fpdf fornece um objeto base e, a partir disso, você constrói um relatório.

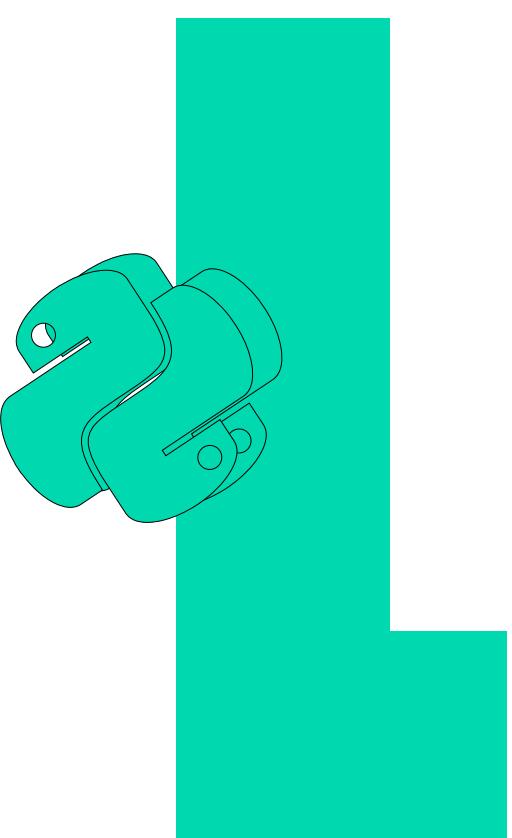
"P" é o formato do PDF, "mm" é a unidade de medida e "Letter" é o tipo da folha.

Sempre que for criar um PDF é necessário adicionar uma página com o add.page e sua fonte.

Você pode entender a criação como se fossem várias "caixas" ou células, por isso temos que sempre colocar o .cell e o tamanho dessa célula.

Dicas:

- Ln pula de linha
- Por padrão, a célula começa na colada no lado esquerdo
- Largura ou altura igual a 0 ocupa a margem inteira
- fill = True, preenche a célula.



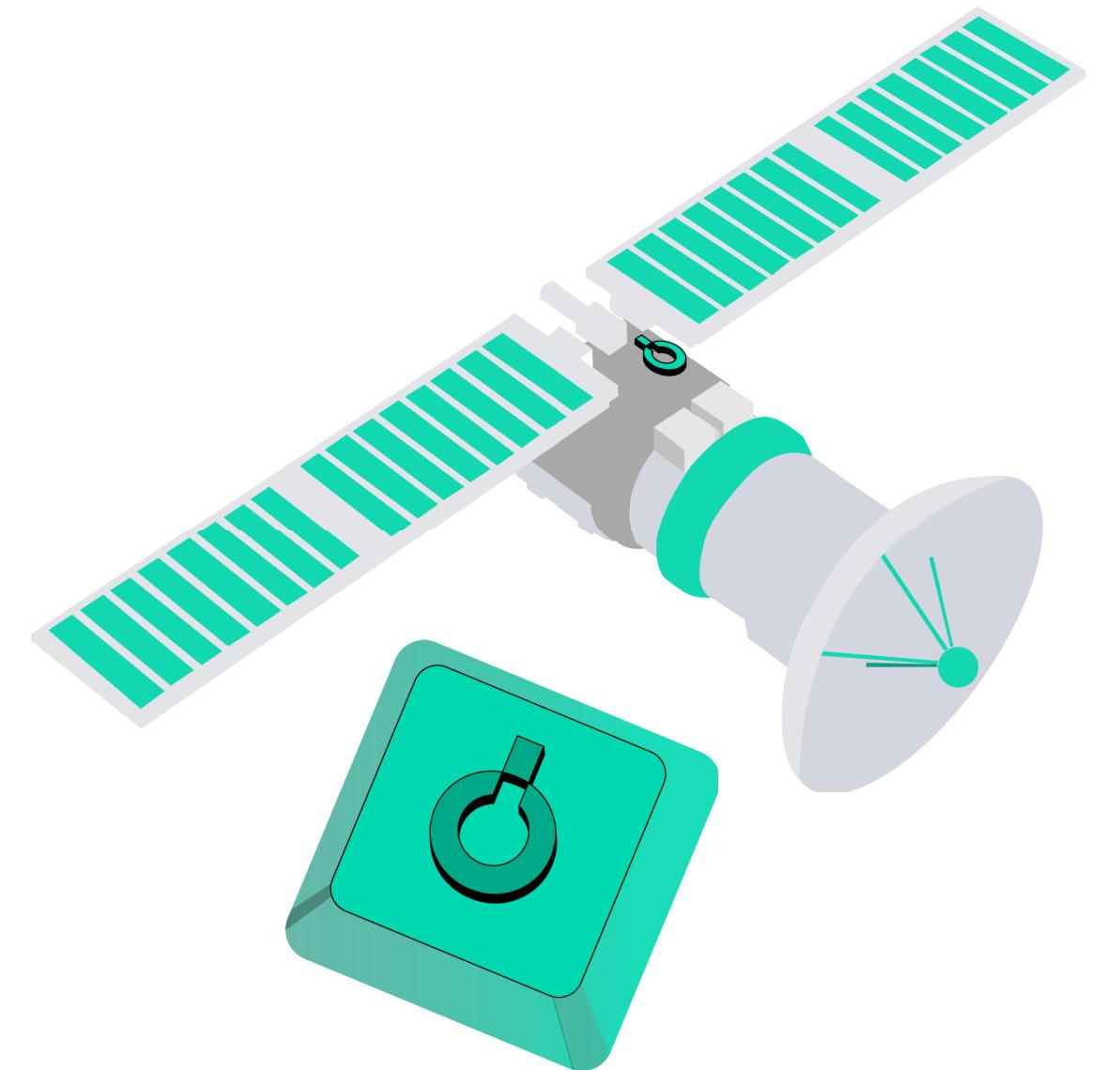
```
pdf = FPDF("P", "mm", "Letter")
pdf.add_page()
pdf.set_font('Arial', 'B', 14)

pdf.cell(0, 50, 'teste', ln = True, border = True,
        fill = False, align = "C")

pdf.cell(10, 10, 'teste2', border = True)
#o alinhamento é do texto

#o tamanho "0" serve como coringa de largura e altura.

pdf.output('mundo12.pdf')
```



Aqui nós vamos trabalhar o conceito de herança nas classes. Nós vamos criar uma nova classe, ou objeto, chamado PDF, que herda todas as características do FPDF e cria alguns padrões próprios de cabeçalho e margem.

Todos os números estão em milímetros.

Nomes importantes:

- Draw color - Cor das linhas.
- Fill color - Preenchimento da célula
- Cell - Célula no PDF, igual ao jupyter notebook.

Coloque a marca d'água e um cabeçalho dentro do header. Lembrando que todas as imagens devem estar dentro da mesma pasta do arquivo.

Com draw_color mude a cor da célula.

Com Footer deixe uma margem no final da página.

No final, configure a numeração das páginas.

```
class PDF(FPDF):  
  
    def header(self):  
  
        self.image('logo.png', 10, 8, 40) #x, y, tamanho  
        self.set_font('Arial', 'B', 20)  
        self.ln(15) #espaço entre a logo e o cabeçalho  
        self.set_draw_color(35, 155, 132) #cor RGB. 0, 0, 0 é preto  
        self.cell(25, ln = False, border = False) #sem isso aqui a margem acaba.  
        self.cell(150, 15, f"Relatório de mercado 01/04/2023",  
                border = True, ln = True, align = "C")  
        self.ln(5)  
  
    def footer(self):  
  
        self.set_y(-15) #espaço ate o final da folha  
        self.set_font('Arial', 'I', 10)  
        self.cell(0, 10, f"{self.page_no()}/{nb}", align = "C")
```

Faça isso para toda vez que acabar uma página, automaticamente adicionar uma nova.

```
#Definindo config básicas do PDF  
pdf = PDF("P", "mm", "Letter")  
pdf.alias_nb_pages()  
pdf.set_auto_page_break(auto = True, margin = 15)  
pdf.set_fill_color(255, 255, 255)  
pdf.set_draw_color(35, 155, 132)  
pdf.add_page()
```

A seguir comece a posicionar as “caixas” no PDF, o processo é o mesmo. Coloque alguns tópicos. É possível também colocar um link de acesso dentro da imagem.

```
pdf.image('navel.png', x = 115, y = 70, w = 75, h = 33, link =
'https://varos.com.br/codigopy')
pdf.set_font('Arial', 'B', 18)
pdf.cell(0, 10, "1 - Ações e câmbio", ln = True, border = False, fill
= False)
pdf.ln(2)

pdf.set_font('Arial', '', 14)
pdf.cell(0, 15, "1.1 Fechamento do mercado", ln = True, border =
False, fill = True)
pdf.ln(7)
```

Os dados aqui foram inventados, quando você for fazer com dados reais antes é necessário criar um programa para pegar esses dados.

```
#fechamento ibov
pdf.set_font('Arial', '', 13)
pdf.cell(25, 15, " Ibovespa", ln = False, border = True, fill = True)
pdf.cell(20, 15, f" 1.95%", ln = True, border = True, fill = False)

#fechamento s&p500
pdf.cell(25, 15, " S&P500", ln = False, border = True, fill = True)
pdf.cell(20, 15, f" -0.65%", ln = True, border = True, fill = False)

#fechamento Dólar
pdf.cell(25, 15, " Dólar", ln = False, border = True, fill = True)
pdf.cell(20, 15, f" 2.11%", ln = True, border = True, fill = False)

pdf.ln(7)
```

Coloque os meses para a tabela de rentabilidade mês a mês.

```
meses = ['Jan', 'Fev', 'Mar', 'Abr', 'Mai', 'Jun', 'Jul', 'Ago', 'Set', 'Out', 'Nov', 'Dez']
```

Faça um loop para criar todas as caixinhas dos meses para não ser repetitivo.

```
pdf.set_font('Arial', '', 14)
pdf.cell(0, 15, " 1.3 Rentabilidade mês a mês", ln = True, border =
False, fill = False)

#escrevendo os meses
pdf.cell(20, 10, "", ln = False, border = False, fill = True, align =
"C")
pdf.set_font('Arial', 'B', 12)

for mes in meses:

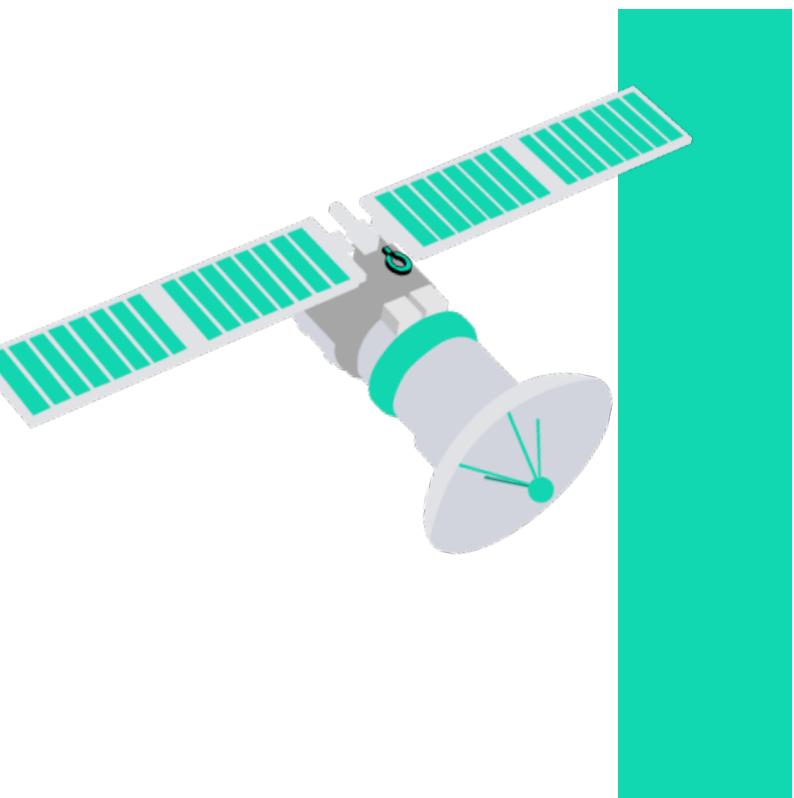
    pdf.cell(15, 10, mes, ln = False, border = True, fill = True,
align = "C")

pdf.ln(10)
```

Baixe o resto do código para poder visualizar completamente, no entanto, não tem nada de novo, no restante o racional é o mesmo.

Lembrando que fazer PDF com Python é muita tentativa e erro, tentar algo, e ver se ficou bom, testar uma fonte, um tamanho, uma largura e ir mudando caso não fique bom.

Então explore conforme as suas necessidades, é possível fazer basicamente tudo com o Python.



Mundo 13

13. Ferramenta 12: Interpolação de taxa de juro - Guia completo de renda fixa no Python

Para começar, pegue a curva de juros, então a primeira parte do código deve ser apenas entrar no site da B3 e pegar a curva de juros de 2019 com web scraping. Pegamos 2019 pois queremos interpolar essa taxa e principalmente, queremos pegar uma curva de juros que seja de um cenário “normal”, mas você pode escolher qualquer data.

Caso tenha dúvida de como pegar esses dados, reveja as aulas da Galáxia 8 de Web Scraping. Então vamos seguir para a segunda parte, sobre Interpolação.

No exemplo vamos ver esse cenário:

Se hoje é dia 06/03/2019.

Qual é o DI de 18/04 de 2023?

Para saber exatamente qual a taxa de juros de uma data deve se fazer uma projeção para interpolar essa taxa. Lembrando que não pode simplesmente fazer uma média, pois isso não funciona com taxas, que são exponenciais.

Você pode fazer esse cálculo com a inflação implícita, que é a diferença do tesouro IPCA e o tesouro pré-fixado de mesmo vencimento.

A inflação implícita por ser marcada no mercado, ou seja, uma amostra muito grande de pessoas operando é muito melhor que apenas a projeção da inflação.

Descubra quantos dias úteis existem entre todas as datas disponíveis de 06/03/2019 e 18/04/2023.

```
curva_dias_uteis = []
dia_atual = datetime(2019, 3, 6)

for dia in tabela.index:

    dias_uteis = len(pd.date_range(dia_atual, dia, freq=BDay()))

    curva_dias_uteis.append(dias_uteis)

curva_dias_uteis
```

Retorne as taxas das datas disponíveis.

```
taxas = tabela.values

taxas = list(taxas)

taxas
```

Resultou em 35 Vértices, que seria o número de datas com taxa divulgada entre as duas datas. Nesse caso, quanto mais vértice melhor, pois o resultado é mais preciso. Para interpolação é necessário sempre no mínimo 3 vértices.

```
len(curva_dias_uteis)
```

Descubra o número de dias úteis apenas de 06/03/2019 e 18/04/2023.

```
dias_uteis_daqui_pra_data = len(pd.date_range(dia_atual, datetime(2023,
4, 18), freq=BDay()))

dias_uteis_daqui_pra_data
```

Importe esse pacote que já existe para fazer o cálculo da interpolação e o conhecido matplotlib para fazer os gráficos.

```
from scipy import interpolate
import mplcyberpunk
import matplotlib.pyplot as plt

plt.style.use("cyberpunk")
```

Faça uma função que descreva a curva da taxa, sendo que essa função pode ser linear ou cúbica. Quando se tem muitos vértices a diferença entre as duas funções é mínima, já quando se tem poucos vértices, a função cúbica normalmente é melhor.

```
linear = interpolate.interp1d(curva_dias_uteis, taxas, kind = 'linear')
cubica = interpolate.interp1d(curva_dias_uteis, taxas, kind = 'cubic')
```

Passe como argumento os dias novos, que deseja calcular e retornando o valor das taxas.

```
dias_novos = [dias_uteis_daqui_pra_data, 520, 385, 2200]
taxas_linear = list(linear(dias_novos))
taxas_cubica = list(cubica(dias_novos))
taxas_linear
```

Plotar o gráfico incluindo os dias novos que não estavam na curva de juros, mas que foi calculado.

```
fig, ax = plt.subplots()

ax.scatter(curva_dias_uteis, taxas)
ax.scatter(dias_novos, taxas_linear)
ax.scatter(dias_novos, taxas_cubica)
```

Mundo 14

14. Ferramenta 13: Como criar uma nuvem de palavras de um resultado de uma empresa

Fazer uma nuvem de palavras pode ser algo muito interessante de fazer para mostrar em uma entrevista de emprego, ou mesmo no seu emprego atual. Você pode fazer isso com basicamente qualquer PDF que tenha texto, desde relatórios, notícias, resultados... entre outros.

Nuvem de palavras é como se fosse um gráfico, então importe o matplotlib e instale também dois novos pacotes, o pypdf e o wordcloud.

```
from pypdf import PdfReader
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
```

A primeira coisa a se fazer é ler um PDF, então escolha o que desejar.

Aqui coloquei como exemplo um de resultados da Weg. O programa vai extrair todo o texto do PDF para verificar as palavras mais citadas.

```
reader = PdfReader("weg.pdf")
text = ""
for page in reader.pages:
    text += page.extract_text()
```

Porém, uma coisa importante é retirar palavras muito comuns, os conectivos conhecidos como StopWords, como por exemplo, "meu", "e" e "você".

```
conectivos = set(STOPWORDS)
conectivos.update(["da", "meu", "em", "e", "você", "de", "ao", "os",
'na', 'o', '4T21', 'para', 'milhões',
'resultado', 'que', 'nas', 'dos', 'n', 'p', '4T22',
"foi", 'ano', 'Klabin', '3T22', 'resultados',
'trimestre', 'weg'])
```

Crie agora a nuvem de palavras.

```
nuvem_palavras = WordCloud(stopwords = conectivos, background_color =  
'black',  
                           width = 1920, height = 1080).generate(text)
```

```
fig, ax = plt.subplots()  
  
ax.imshow(nuvem_palavras)  
ax.set_axis_off()
```

Mundo 15

15. Ferramenta 14: Análise de sentimento de um texto com o Python

Para fazer uma análise de sentimento vamos precisar o ChatGPT para traduzir textos do portugues para o inglês e depois a resposta, já que a ferramenta que vamos usar no Python, só consegue ler textos em inglês.

O site da OpenAI tem vários modelos além do ChatGPT que podem ser usados e que valem a pena ser explorados, leiam a documentação.

<https://platform.openai.com/docs/api-reference>

Após importar os pacotes e abrir duas notícias que possuem sentimentos diferentes, faça o web scraping, que novamente, já foi explicado na Galáxia 8, em caso de dúvidas no código.

Uma vez que isso foi feito, pegue o texto e envie para o ChatGPT.

É necessário se logar no site da OpenAI e entrar na parte de chaves de API.

<https://platform.openai.com/api-keys>

Após criar uma nova chave, coloque em uma .env.

```
load_dotenv()
```

Coloque uma variável para pegar essa chave.

```
token = os.environ.get("openai_token")
```

Como cabeçalho, faça um Authorization e um Content-Type para transformar o texto no formato json.

```
headers = {'Authorization': token,  
           'Content-Type': 'application/json'}
```

Faça Request para ver os nomes dos modelos.

```
response = requests.get('https://api.openai.com/v1/models', headers=  
headers)  
  
pprint(response.json())
```

Escolha o gpt-3.5-turbo, e é válido mencionar que se algum dia essa versão mudar, é só verificar e alterar o modelo aqui.

```
id_modelo = 'gpt-3.5-turbo'

texto_traduzido = []
```

Aqui percorra os textos na lista de parágrafos e envie uma mensagem para o Chat GPT. Faça uma requisição a cada parágrafo e retorne um json. Armazene a resposta em uma lista nova.

```
for texto in paragrafos:

    mensagem_gpt = {'model': id_modelo,
                    'messages': [{'role': 'user',
                                  'content': "Traduza o seguinte texto
para o inglês, me retornando APENAS a tradução:" + texto.text}]}

    json_mensagem = json.dumps(mensagem_gpt)

    response =
    requests.post('https://api.openai.com/v1/chat/completions',
                  headers = headers, data =
    json_mensagem)

    texto_traduzido.append(response)
    time.sleep(15)
```

Faça o código para o texto final, onde o código 200 significa que está tudo certo. Retorne o texto completo e traduzido.

```
texto_final = ''

for texto in texto_traduzido:

    if texto.status_code == 200:

        parte_traduzida =
        texto.json()['choices'][0]['message']['content']

        texto_final += parte_traduzida

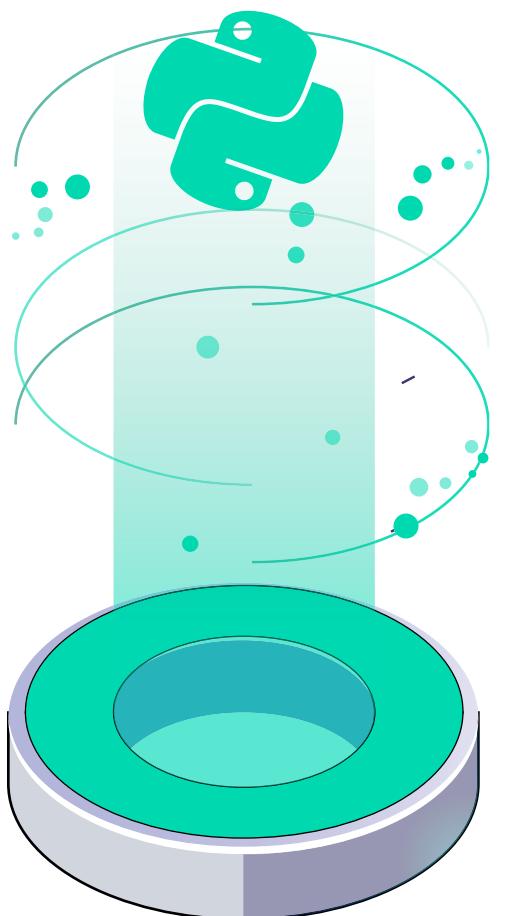
    texto_final
```

Agora finalmente, faça a análise de sentimentos.

```
blob = TextBlob(texto_final)
```

O espectro vai de -1 a 1, onde 1 seria um texto muito feliz e -1 muito crítico.

```
sentimento = blob.sentiment.polarity  
sentimento
```



Mundo 16

16. Ferramenta 15: Como ler uma tabela dentro de um PDF

Nessa aula utilize o pacote pypdf2 e o tabula, diferente do que foi usado anteriormente.

Para usar o tabula é necessário ter o Java instalado no computador. Você pode verificar se possui apenas pesquisando "Java" no seu computador. Caso não tenha, para instalar é muito simples, pesquise no Google e baixe, mas lembre de reiniciar a IDE que estiver usando, seja o Jupyter Notebook ou VScode, depois.

```
import PyPDF2  
import re  
import tabula
```

É importante mencionar também, que extrair dados de um PDF não é algo tão trivial de se fazer pois ele não é feito para isso, é uma ferramenta de visualização apenas.

```
pdf_file = open('weg.pdf', 'rb')

#Faz a leitura usando a biblioteca
pdf = PyPDF2.PdfReader(pdf_file)

# pega o numero de páginas
paginas = len(pdf.pages)

paginas
```

Extraia o texto da primeira página.

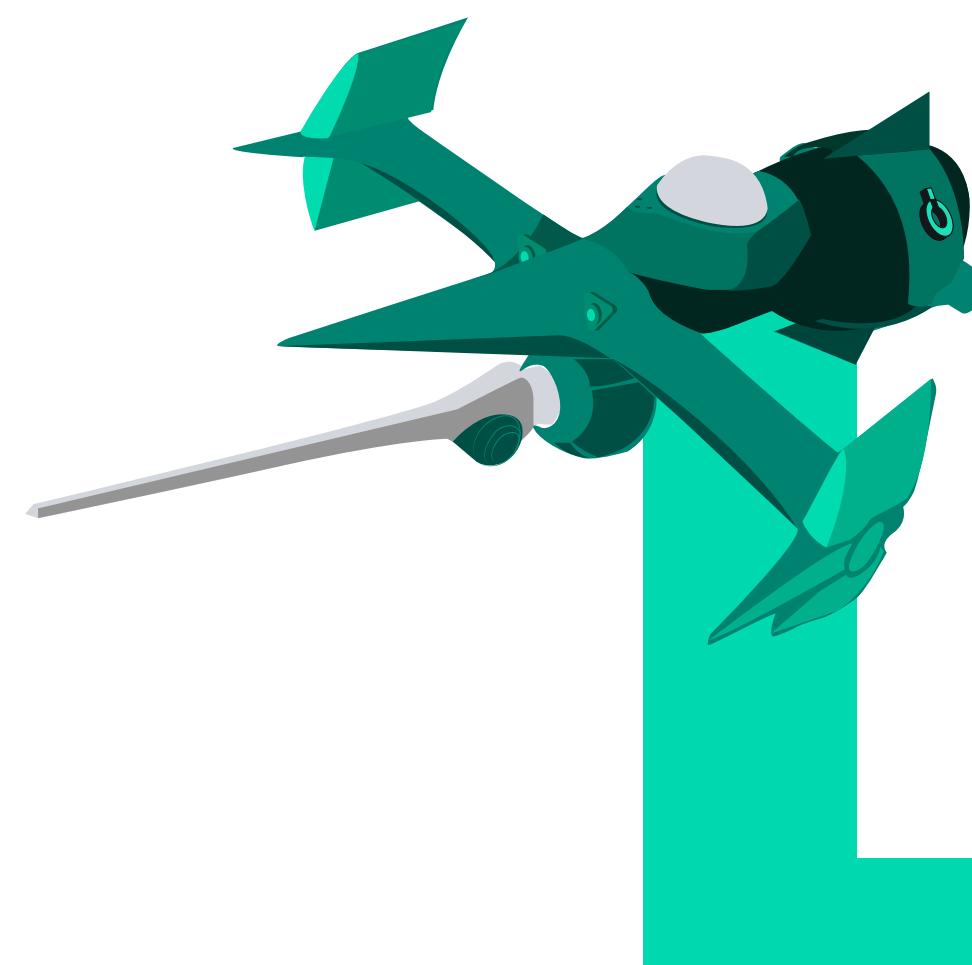
```
#lê a primeira página completa
pagina1 = pdf.pages[0]

pagina1.extract_text()
```

Caso queira formatar melhor você pode apenas dar um .join

```
# faz a junção das linhas
texto_formatado = ''.join(pagina1.extract_text())

print(texto_formatado)
```



Para extração de tabelas. Os argumentos Guess = False e Lattice = True são usados na tentativa de obter tabelas ou dados escondidos, mas não são garantias que você vai conseguir.

```
tabelaComum = tabula.read_pdf("klabin.pdf",
                               pages = "3", lattice = True, guess =
                               False) #guess = False e lattice = True
```

Converta as tabelas em DataFrame.

```
for tabela in tabelaComum:
    display(tabela)
```

A partir daqui você com o que já aprendeu de Python e pandas, pode formatar essas tabelas e deixar do jeito que preferir.

Mundo 17

17. Ferramenta 16: Criando um ambiente virtual no Python - Fundamental para sistemas complexos

Criar um ambiente virtual no Python é imprescindível, esse ambiente é como se fosse uma bolha onde se está trabalhando somente com o que precisa para rodar o programa, onde você pode otimizar memória, espaço e processamento.

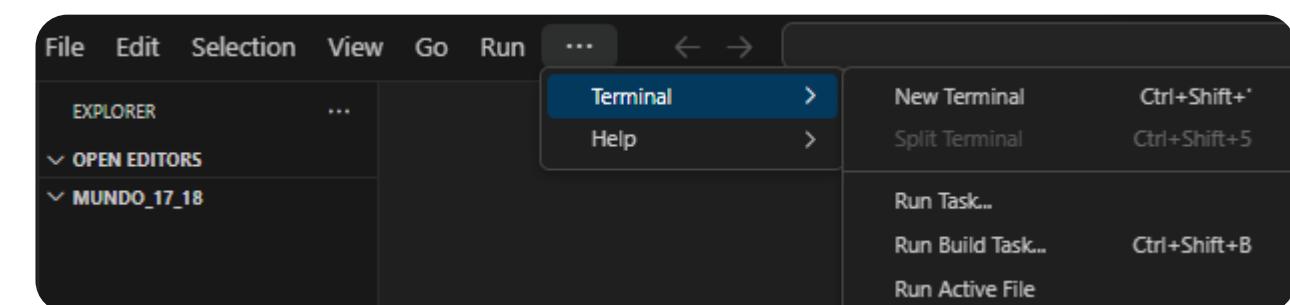
Para descobrir onde o Python está, você pode abrir o prompt de comando e digitar: "where python". Caso queira saber todos os pacotes e suas versões que tenha no seu computador, basta digitar "pip freeze".

Ou seja, o seu ambiente de execução do Python já possui todos esses pacotes, deixando ele muito pesado. Criar um ambiente virtual com somente os pacotes que precisa ajuda nessa questão deixando bem mais leve. Que é o ideal quando for criar uma ferramenta grande de fato.

Criar esse ambiente também ajuda quando você precisa rodar dois programas todos os dias em duas versões de pacotes diferentes, algo impossível quando se trabalha no ambiente tradicional. Também é muito útil para rodar em nuvem como AWS, pois ocupa menos espaço e menos recursos, ou seja, menos gastos.

Como criar?

No VSCode, crie uma nova pasta onde você costuma fazer seus projetos e dentro dessa pasta nova vazia, crie um terminal.



No terminal, digite o seguinte comando: **py -3 -m venv .venv**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
PS C:\Users\lsiqu\dev\github\codigos_curso\galaxia_9_ferramentas_fin\mundo17_18> py -3 -m venv .venv
PS C:\Users\lsiqu\dev\github\codigos_curso\galaxia_9_ferramentas_fin\mundo17_18> [ ]
```

Após esse passo, entre no windows PowerShell, ou terminal, do seu computador. Importante é ser a versão de administrador. Digite o seguinte comando para autorizar o Windows a criar um ambiente virtual:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

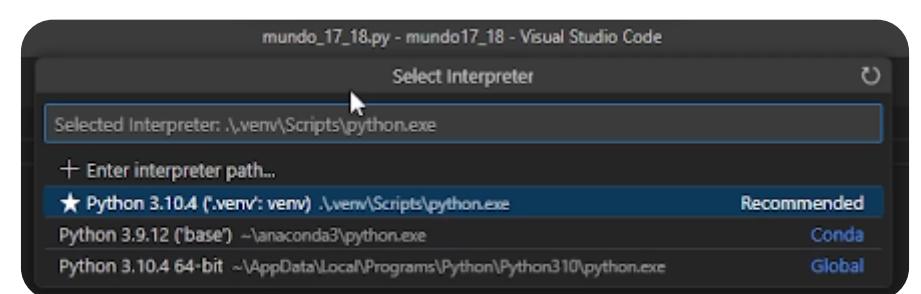
Alteração da Política de Execução
A política de execução ajuda a proteger contra scripts não confiáveis. A alteração da política de execução pode implicar alterar a política de execução?
[S] Sim [A] Sim para Todos [N] Não [T] Não para Todos [U] Suspender [?] Ajuda (o padrão é "N"): s
```

Existem outras maneiras de se criar um ambiente virtual, mas essa é a mais simples.

Para acessar o ambiente dentro do VSCode, digite o comando:
.venv\scripts\activate

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
PS C:\Users\lsiqu\dev\github\codigos_curso\galaxia_9_ferramentas_fin\mundo17_18> py -3 -m venv .venv
PS C:\Users\lsiqu\dev\github\codigos_curso\galaxia_9_ferramentas_fin\mundo17_18> .venv\scripts\activate
(.venv) PS C:\Users\lsiqu\dev\github\codigos_curso\galaxia_9_ferramentas_fin\mundo17_18> [ ]
```

Agora para rodar qualquer programa, lembre-se de selecionar o Python .venv.

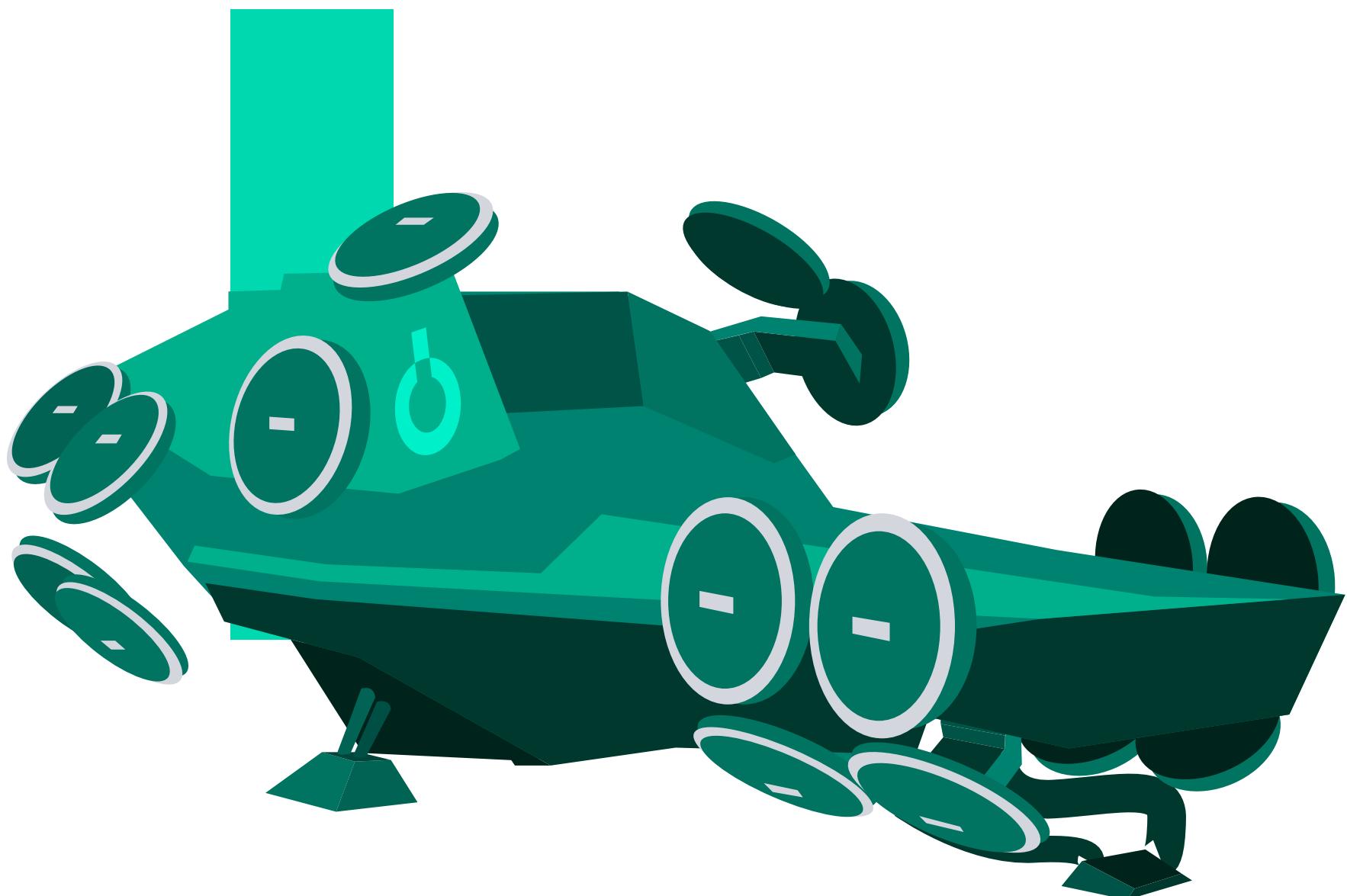


No Linux e MacOS:

No seu terminal, entre na área de trabalho e digite: **python -m venv venv**

Uma vez criado seu ambiente virtual, você deve ativá-lo, com o comando: **source venv/bin/activate**

<https://docs.python.org/pt-br/3/tutorial/venv.html>



Mundo 18

18. Ferramenta 17: Como transformar um arquivo Python em Executável - Rodando automações em computadores sem o Python baixado

Dando sequência a aula anterior, aqui vamos aprender como fazer um arquivo executável no Python, mas por que isso? Ter um executável é essencial caso você queira enviar seus projetos e trabalhos para outras pessoas que não possuem ou não tem ideia de como mexer no Python, para que elas consigam utilizar o programa.

Para criar um executável é essencial que tenha um ambiente virtual, então lembre de criar um antes de começar.

Então dentro do ambiente virtual, comece digitando: **pip install pyinstaller**

Após isso, o comando: **pyinstaller --onefile -w mundo_17_18.py**

Esse “-w” no comando é importante em programas que têm interação com o usuário, caso o seu não possua, não é necessário colocar.

O "mundo_17_18.py" é o nome do programa, aqui mude para o seu.

Pronto! Seu código virou um aplicativo que pode ser enviado para qualquer pessoa.

Alguns pontos que devem ser considerados:

Programas que precisam de senhas, como envio automático de e-mails podem ser resolvidos de duas maneiras.

Primeiro, você pode colocar um input e toda vez que a pessoa for rodar esse executável ela vai ter que digitar sua senha para o arquivo funcionar. Pode ser mais "chato" mas é muito mais seguro.

Segundo, caso queira 100% automatizado será necessário colocar a senha dentro do código. No entanto, caso a pessoa mude de senha ou esse arquivo vaze para alguém que entenda de programação, é possível reverter o executável para código, e a senha estará exposta.

Outro ponto são programas que abrem planilhas, é importante que essa planilha esteja na mesma pasta do programa da pessoa que for executar para que dê tudo certo.

Essa foi a galáxia 9, conseguimos aprender muitas ferramentas úteis e legais que podem ser utilizadas no dia a dia, e na próxima galáxia vamos aprender finalmente sobre modelos quantitativos!