tree.txt

```
FisComp.py/
��� ComecandoAProgramar.md
��� exemplos/
�   ��� lacamento_arrasto/
�   �   ��� lanc.py
�   ��� logistic_map/
�   �   ��� log_map.py
�   ��� mandelbrot_set/
�       ��� mandel.py
��� intro/
�   ��� 00hello_world.py
�   ��� 01inputs.py
�   ��� 02operacoes.py
�   ��� 03funcoes.py
��� LICENSE
��� notas_de_aula/
�   ��� notas_2024_1/
�       ��� aula_02_abr.md
�       ��� aula_14_mar.md
�       ��� aula_21_mar.md
�       ��� codigos_alunos/
�       �   ��� daniel_21_03_2024_baskara.py
�       ��� tarefas/
�           ��� desvendando_os_primos.md
�           ��� fig_x2.png
�           ��� graficos.md
��� README.md
```

01inputs.py

```python
FisComp.py/intro/01inputs.py
# Nesse programa vamos mostrar
# como escrever inputs
# e inporá-los em um programa

nome_do_aluno = input("Por favor, digite seu nome...")

mensagem = nome_do_aluno+" é um aluno da disciplina de Computação Básica para Física!"
```

03funcoes.py

```python
FisComp.py/intro/03funcoes.py
# Em Python podemos escrever uma função para
# realizar tarefas repetitivas para nós
# digamos que queremos calcular o valor da
# hipotenusa de um triângulo retângulo
# dados dois lados, l1 e l2.
from math import sqrt

def hipotenusa(l1,l2):
    # A hipotenusa ao quadrado é igual à soma
    # dos quadrados dos catetos
    #
    # A raiz quadrada da soma dos quadrados
    # a sqrt() é uma função que calcula a raiz quadrada
    # do que está entre os parênteses
    # e quando um número, x, é elevado ao quadrado, precisamos
    # escrever x**2.
    # por exemplo, 2 ao quadrado é escrito como 2**2, que é igual a 4.
    return sqrt(l1**2 + l2**2)

if __name__ == "__main__":
    print("Para um triângulo retângulo com lados 3 e 4 a hipotenusa é igual à:")
    print(hipotenusa(3,4))
```

02operacoes.py

```python
FisComp.py/intro/02operacoes.py
# Solicita ao usuário dois números
num1 = float(input("Digite o primeiro número: "))
num2 = float(input("Digite o segundo número: "))

# Adição
soma = num1 + num2
print(f"{num1} + {num2} = {soma}")

# Subtração
subtracao = num1 - num2
print(f"{num1} - {num2} = {subtracao}")

# Multiplicação
multiplicacao = num1 * num2
print(f"{num1} * {num2} = {multiplicacao}")

# Divisão
divisao = num1 / num2
print(f"{num1} / {num2} = {divisao}")

# Divisão inteira
divisao_inteira = num1 // num2
print(f"{num1} // {num2} = {divisao_inteira}")

# Módulo (resto da divisão)
modulo = num1 % num2
print(f"{num1} % {num2} = {modulo}")

# Potenciação
potencia = num1 ** num2
print(f"{num1} ** {num2} = {potencia}")
```

ComecandoAProgramar.md

```markdown
FisComp.py/ComecandoAProgramar.md
# Começando a programar

Nesse arquivo est��o listados alguns recursos para programar em python.

Muitas vezes a primeira barreira para a programa����o pode ser o fato de você nunca ter sido instruído sobre onde escrever o programa e como execut��-lo.
De forma simples, um programa em python pode ser escrito em um editor de texto e executado com uma linha de comando no terminal.

# Alguns conceitos importantes

### Editor de texto

No editor de texto você pode escrever um programa em python e salv��-lo na extens��o `.py`.
Um programa é um conjunto de instru����es (algoritmo) escrito e estruturado de uma forma específica (sintaxe da linguagem) para ser executada ou compilada depois.

### Terminal

Podemos executar um programa no terminal utilizando o comando:
bash```python3 programa.py```

O terminal é um ambiente de input e output de dados. Os dados de input s��o os comandos e os outputs podem ser a execu����o dos programas ou a informa����o requisitada.
De forma simples, é uma forma de executar processos no computador através de comandos de texto.

## WSL - Usando Linux no Windows

Caso o seu sistema operacional seja Windows 10 ou 11, como é no meu caso, é possível utilizar o WSL (Windows Subsistem for Linux).
As instu����es do [site oficial da Microsoft](https://learn.microsoft.com/pt-br/windows/wsl/install) podem ser úteis.

## Virtual Machine

Outra possibilidade é a utiliza����o de uma m��quina virtual que emula outro sistema operacional para utiliza����o dos programas em Linux.
Programas como [VirtualBox](https://www.virtualbox.org/) podem ser instalados para isso.

### REPL

�� possível programar em python diretamente do terminal, utilizando o REPL.

```

```
O REPL é uma sigla para "Read,Evaluate,Print,Loop", ou seja, ele lê, executa, imprime e espera o seu próximo comando para fazer tudo isso novamente.
__A palavra "loop" ou "laço" aparece em um programa quando queremos dizer que um certo "pedaço" do programa vai ser repetido mais de uma vez.__

No REPL você escreve um programa linha por linha e ele vai sendo executado em tempo real. Esse tipo de execu����o de programas é melhor para testes r��pidos de funcionalidades do que

### IDE

Um IDE, ou Integrated Development Environment (ambiente integrado de desenvolvimento).
�� um programa que combina no mesmo ambiente um editor de texto e ferramentas para execu����o do programa a ser desenvolvido, como um terminal integrado, na maioria dos casos.
Exemplos de IDEs:

- VSCode
- IDLE
- Atom
- PyCharm

Para poder executar python localmente em seu computador, é necess��rio instalar.

## Python.org

No [site oficial da linguagem python](https://www.python.org/) é possível baixar o interpretador de python para o seu computador pessoal.

Em um computador com sistema linux podemos instalar python pelo terminal com o comando
bash```sudo apt-get install python3```

Também é possível executar blocos de programas em python na nuvem do Google, no site Google Collab.

## Google Collab

No [Google Collab](https://colab.research.google.com/) temos acesso a um notebook que interpreta python online.
Muitas bibliotecas est��o disponíveis e a colabora����o entre diferentes colaboradores é facilitada.

Para come��ar a testar novos códigos, podemos utilizar o google collab como mostrado nesse [exemplo em vídeo](https://www.youtube.com/watch?v=ww0IcgQp8vI).

## Compilador online de matplotlib
[Nesse link](tutorialspoint.com/execute_matplotlib_online.php) podemos acessar um recurso online que plota gr��ficos usando o matplotlib diretamente.
```

00hello_world.py

```python
FisComp.py/intro/00hello_world.py
# Nesse programa vamos fazer o que todo
# tutorial de python faz, que é printar
# uma mensagem para mostrar como podemos
# mostrar mensagens no terminal
# <- a propósito isso aqui é um comentário
print("Hello World, ou Olá mundo, já que estamos no Brasil!")
```

log_map.py

```python
FisComp.py/exemplos/logistic_map/log_map.py
import numpy as np
import matplotlib.pyplot as plt

# Definindo os parâmetros iniciais
r_values = np.linspace(2.5, 4.0, 10000)
iterations = 1000
last = 100

x = 1e-5 * np.ones(10000)

# Criando o gráfico
plt.figure(figsize=(10, 6))

# Executando as iterações do mapa logístico
for i in range(iterations):
    x = r_values * x * (1 - x)
    if i >= (iterations - last):
        plt.plot(r_values, x, ',k', alpha=0.25)

plt.title("Mapa Logístico")
plt.xlabel("Taxa de Reprodução (r)")
plt.ylabel("População (x)")
plt.xlim(2.5, 4.0)
plt.ylim(0, 1)
plt.show()
```

mandel.py

```python
FisComp.py/exemplos/mandelbrot_set/mandel.py
import numpy as np
import matplotlib.pyplot as plt

def mandelbrot(c, max_iter):
    z = 0
    n = 0
    while abs(z) <= 2 and n < max_iter:
        z = z*z + c
        n += 1
    return n

def main():
    # Tamanho da imagem (em pixels)
    largura, altura = 800, 600

    # Limites do plano complexo que queremos visualizar
    x_min, x_max = -2, 1
    y_min, y_max = -1.5, 1.5

    # Cria um array para armazenar os valores dos pixels
    imagem = np.zeros((altura, largura))

    # Define o número máximo de iterações
    max_iter = 100

    # Gera os valores para cada pixel
    for i in range(altura):
        for j in range(largura):
            c = complex(x_min + (x_max - x_min) * j / largura, y_min + (y_max - y_min) * i / altura)
            imagem[i, j] = mandelbrot(c, max_iter)

    # Mostra a imagem
    plt.imshow(imagem, cmap='hot', extent=(x_min, x_max, y_min, y_max))
    plt.colorbar()
    plt.title("Conjunto de Mandelbrot")
    plt.xlabel("Re(c)")
    plt.ylabel("Im(c)")
    plt.show()

if __name__=="__main__":
    main()
```

lanc.py

```python
FisComp.py/exemplos/lacamento_arrasto/lanc.py
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt
# Constantes
g = 9.81  # aceleração devido à gravidade (m/s^2)
b = 0.1   # coeficiente de resistência do ar (proporcional ao quadrado da velocidade)

# Condições iniciais
v0 = 30  # velocidade inicial (m/s)
angle = 45  # ângulo de lançamento em graus
vx0 = v0 * np.cos(np.radians(angle))  # velocidade inicial x
vy0 = v0 * np.sin(np.radians(angle))  # velocidade inicial y
y0 = 0  # posição inicial y (altura)
x0 = 0  # posição inicial x

# Equações do movimento
def motion(t, Y):
```

```python
    x, vx, y, vy = Y
    dvx_dt = -b * vx * np.sqrt(vx**2 + vy**2)
    dvy_dt = -g - b * vy * np.sqrt(vx**2 + vy**2)
    dx_dt = vx
    dy_dt = vy
    return [dx_dt, dvx_dt, dy_dt, dvy_dt]

# Intervalo de tempo para a simulação
t_span = (0, 3)  # tempo total de 5 segundos
t_eval = np.linspace(*t_span, 1000)  # 1000 pontos de tempo para avaliação

# Resolvendo as equações de movimento
sol = solve_ivp(motion, t_span, [x0, vx0, y0, vy0], t_eval=t_eval, method='RK45')

# Plotando o gráfico da trajetória
plt.figure(figsize=(10, 6))
plt.plot(sol.y[0], sol.y[2], label='Trajetória com Atrito')
plt.title("Lançamento Oblíquo com Atrito")
plt.xlabel("Distância Horizontal (m)")
plt.ylabel("Altura (m)")
plt.legend()
plt.grid(True)
plt.show()
```

aula_21_mar.md

```markdown
FisComp.py/notas_de_aula/notas_2024_1/aula_21_mar.md
# Notas de aula do dia 21 de Março de 2024

## Sintaxe

A sintaxe de uma linguagem é a forma como escrevemos.
A sintaxe se preocupa como os elementos de uma frase, ou de uma linha de código, é escrita.
Nas linguagens de programa����o a sintaxe possui regras que s��o seguidas pelo interpretador ou pelo compilador para executar o programa da forma como ele foi __escrito__.
A forma como o programa é __escrito__ é como ele vai ser executado, nem sempre ele vai ser executado da forma desejada, isso ocorre quando a sintaxe n��o est�� correta, ou est�� corre

Um bom entendimendo da sintaxe é necess��rio para entedermos como um programa funciona e como podemos "traduzir" as nossas ideias, ou os algoritmos que queremos executar, para um prog

### Comentários

Em linguagens de programa����o é comum que se tenha diferentes formas de escrever coment��rios nos códigos.
Os coment��rios s��o ignorados pelo compilador ou interpretador da linguagem e servem apenas para fazer anota����es no programa.
Eles podem ter diferentes funcionalidades, como explicar o que cada linha do programa faz ou para servir de guia para próximas pessoas que forem utilizar os programas caso eles precis
�� muito importante escrever coment��rios relevantes que possam ajudar pessoas no futuro a entender os seus programas, muitas vezes essa pessoa do futuro pode ser __você__.

## Variável

Vari��veis s��o símbolos que guardam informa����o. �� comum escrevermos programas que funcionem para calcular quantidades, por exemplo, ou para trocar mensagens.
Quais s��o essas quantidades ou quais s��o essas mensagens nós n��o sabemos, essas coisas v��o mudar toda vez que o programa for executado.
Mesmo assim a lógica do que fazemos com essas quantidades vai ser sempre a mesma, ent��o podemos escrever um programa que funcionaria para qualquer valor.

A opera����o de "pegar" um valor e "guardar" ou "escrever" ele em um símbolo é chamado de atribui����o.
Em python o operador de atribui����o é o símbolo `=`, que n��o pode ser utilizado para guardar informa����o nenhuma. Nenhum símbolo de opera����o pode ser utilizado para guardar outra

Ent��o se quisermos guardar uma vari��vel que é aproximadamente o número $\pi$ podemos fazer isso da seguinte maneira:
```
pi = 3.1415926
```
O que isso significa é que na vari��vel chamada `pi` est�� guardado o número do tipo "float" que é igual a `3.1415926`.
Toda vez que essa vari��vel `pi` aparecer em algum lugar, ela se comportar�� como o número que est�� guardado nela.

Essas ideias s��o muito poderosas e podem ser utilizadas para criar programas que realizam diferentes tarefas.

### Tipos

Para podemos utilizar uma linguagem de programa����o para resolver problemas reais precisamos poder realizar opera����es entre quantidades, como números ou texto.
A informa����o existe em diversos tipos, um número de telefone pode ser um número inteiro, mas pode ser melhor representado como um texto, devido aos outros caracteres, por exemplo.

> O que é mais f��cil de ler?
> `5555999991111` ou `"+55 (55) 99999 1111"`?

Dependendo do contexto a informa����o pode ser representada de diferentes formas, e os diferentes tipos s��o uma forma de classificar diferentes tipos de informa����o.


#### Tipos numéricos

Nessa matéria vamos utilizar diferentes valores

- float: "floating point number" ou "número com algarismo decimal" (ou "real"), é uma forma de representar números com diferentes casas depois da vírgula.
- - Bom para c��lculos numéricos que exigem precis��o.
- - Escrevemos números float com `.` como separador decimal e n��o `,`.
- - Por exemplo `0.1254` ou `3.1415926`.

- int: "integer" ou "número inteiro", é uma forma de representar números inteiros,
- - Bom para contagem ou coisas que tomam valores inteiros.
- - Escrevemos números inteiros sem `.`.
- - Como por exemplo: `12` ou `1917`.

Diferentes opera����es entre tipos numéricos podem levar a resultados diferentes em diferentes linguagens, é importante testar elas antes de colocar em seu programa.
Uma forma útil de fazer isso é utilizando a fun����o `print()` com o valor dentro, mostrando o valor na tela.

Por exemplo, em python 2 temos como resulado da divis��o entre dois números inteiros um número inteiro.
Em python 3 o resultado pode ser tanto um inteiro como um float. Experimente!

#### Tipo "string" ou "frase"

O tipo string é um tipo que pode ser tanto um caracter ou uma concatena����o de caracteres. Nem todos os caracteres s��o suportados por todas as linguagens.
Em python 3 podemos utilizar praticamente todos os caracteres presentes no teclado.

Vari��veis do tipo string podem ser utilizadas para mostrar informa����es ou para guardar informa����es em formato de texto.

Uma string em python é definida como o conjunto de caracteres que é escrito entre aspas ('') ou aspas duplas ("").

Por exemplo `"isso é uma string"` e `'isso também é uma string'`.
```

aula_02_abr.md

~~~markdown
FisComp.py/notas_de_aula/notas_2024_1/aula_02_abr.md
# Aula do dia 02 de Abril de 2024

Nessa aula a gente reviso alguns conceitos b��sicos de fun����es e como criar e popular vetores (Arrays) em python utilizando loops "while".

Um loop while funciona como um loop de repeti����o qualquer, que vai executar o que estiver contido em seu interior (dentro do 'parágrafo') até que a condi����o fornecida seja falsa. 
```python
while (condi����o):
        (código a ser repetido)
~~~

Podemo por exemplo utilizar o programa para popular um vetor vazio.
Digamos que queremos preencher um vetor vazio com valores que come��am em `0.0`até `100.0` com passo arbitr��rio. Podemos fazer isso da seguinte maneira:
```python
vetor = [] # define vetor vazio
x0 = 0.0 # variável de início
passo = 0.25 # define passo arbitrário

while x0 < 100.0:
        vetor.append(x0)
        x0 = x0 + passo
```

Ao executar esse programa vamos ter armazenado na vari��vel `vetor` um vetor com as componentes iniciando em `0.0` aumentando em passos de `0.25`.

Plotando gráficos

Podemos utilzar esse conhecimento de criar vetores para criar pontos do gr��fico de uma fun����o. Por exemplo para a fun����o $f(x) = x^2$ podemos criar um outro vetor vazio e colocar
```
```

```
FisComp.py/notas_de_aula/notas_2024_1/aula_14_mar.md
# Bem vindos ao curso!

No curso de computa����o b��sica para física vamos estudar sobre o que é computa����o, o que é programa����o e como programar.
Na primeira aula vamos conversar sobre o que é computa����o.

## O que é computação?

A computa����o surgiu da matem��tica.
Computar significa "calcular" e por muitos anos os computadores eram pessoas.
Por suas vezes, grupos de pessoas respons��veis por fazer contas extensas, como multiplica����o de números muito grandes "na mão" com ferramentas b��sicas como "ábacos" ou outras mais
Exemplos de pessoas reais que trabalhavam como computadoras s��o as mulheres representadas no filme "Estrelas além do tempo" (recomenda����o minha, ótimo filme).

> Nesse filme as mulheres que trabalhavam como computadores est��o prestes a serem substituídas
> por um computador digital da IBM (IBM 7090) que compila programas em FORTRAN, linguagem de programa����o tradicionalmente utilizada
> na disciplina correlata a essa na ementa anterior do curso de Física.

Hoje em dia podemos realizar c��lculos utilizando softwares (programas desenvolvido por pessoas) que funcionam em harware (pe��as e componentes físicos).
Os objetos aos quais nos referimos hoje como computadores pessoais, ou "PC" (Personal Computer), s��o conjuntos de hardwares e softwares que trabalham juntos para realizar diferentes
O seu celular também é um computador (muito mais poderoso que o IBM 7090, que foi utilizado para levar alguns dos primeiros seres humanos ao espa��o).

### Como funciona o computador?

Um computador possui o hardware e o software:

> O hardware tu chuta, o software tu xinga.
> -- Autor desconhecido

#### Hardware

As parte físicas de um computador s��o chamadas de hardware:
- Placa M��e: Placa de circuitos que liga todas as outras pe��as de hardware.
- Memória RAM (random-access memory): Memória de "curto prazo".
    - A memória RAM é respons��vel por guardar os dados que est��o sendo processados no momento.
    - Ela e "limpa" toda vez que o computador é desligado.
    - Ela possui acesso mais r��pido que o HD e o SSD (na maioria das vezes).
- HD (Hard Drive - disco rígido) ou SSD (Solid State Drive - drive de estado sólido):
    - Eles s��o a memória de "longo prazo".
    - Utilizados para armazenar os dados e arquivos que s��o salvos.
    - SSDs s��o mais r��pidos que HDs.
- CPU (Central Processing Unit - unidade de processamento central - "processador"):
    - Unidade central de processamento do computador.
    - �� respons��vel por coordenar os processos do computador e realizar as computa����es.
- GPU (Graphics Processing Unit - unidade de processamento gr��fico - "placa de vídeo").

#### Software

Um software é um programa, programas podem ser escritos em diferentes linguagens para realizar diferentes tarefas.
Nessa disciplina vamos buscar entender como entender a lógica por tr��s dos algoritmos para podermos escrever bons softwares para realizarmos diferentes atividades.
Exemplos de softwares necess��rios para utilizar um computador s��o os seguintes:
- Kernel: O kernel é um software que é respons��vel por controlar todos os processos do computador, ele é a pe��a fundamental que permite que as diferentes pe��as (hardware) trabalhem
    - O kernel é a parte principal do sistema operacional, respons��vel pela usabilidade do computador.
- Sistema Operacional (OS ou SO):
> Conceito: �� o conjunto de programas que gerenciam recursos, processadores, armazenamento, dispositivos de entrada e saída e dados da m��quina e seus periféricos. O sistema que faz

(notas incompletas - terminarei elas em algum momento)
```

```
FisComp.py/notas_de_aula/notas_2024_1/tarefas/fig_x2.png
�PNG



IHDR�j��y9tEXtSoftwareMatplotlib version3.8.3, https://matplotlib.org/��XS      pHYs��n�u>��IDATx���w�\�6��lO�$$!
```

[illegible]

graficos.md

~~~
FisComp.py/notas_de_aula/notas_2024_1/tarefas/graficos.md
# Mostrando gráficos de funções

Utilizando a biblioteca matplotlib, escreva dois vetores, um com os dados das coordenadas $x$ e outros com as coordenadas $y$ e "plote" os gr��ficos das seguintes fun����es:
- $\tan(x)$ no intervalo $x \in [-\pi,\pi]$
- $\cos(x^2)$ no intervalo $x \in [-\pi,\pi]$
- $x^3-x$ no intervalo $x \in [-2,2]$
- $x^x$ no intervalo $x \in [0,1]$
- $e^{-x^2}$ no intervalo $x \in [-3,3]$
- $x^2 e^{-x}$ no intervalo $x \in [-1,5]$


## Programa para funções

Instru����es:

- Tenha instalaldo o pacote `matplotlib.pyploy` em sua m��quina.
- Escreva um programa que popule duas vari��veis do tipo array com os valores desejados de $x$ e $y$, por exemplo `x=[0,1,2,3,4]` e `y=[0,1,4,9,16]` para uma fun����o do tipo quadr��t
- Em um arquivo escreva um programa como:
```python
import matplotlib.pyplot as plt

x = [0,1,2,3,4]
y = [0,1,4,9,16]

plt.plot(x,y)
plt.show()
```

Esse programa vai gerar uma imagem como essa ![image](fig_x2.png).
~~~

desvendando_os_primos.md

~~~
FisComp.py/notas_de_aula/notas_2024_1/tarefas/desvendando_os_primos.md
# Desvendando a função de números primos

Temos aqui uma fun����o que nos diz se um número `n` é primo ou n��o.

```python
def is_prime(n):
    if n <= 1:
        return False
    elif n <= 3:
        return True
    elif n % 2 == 0 or n % 3 == 0:
~~~

```
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
```

Essa fun����o tem como entrada um número `n` e tem como saída um valor booleano de `True` ou `False`.

A partir da sintaxe da fun����o e das opera����es que ela executa, explique as seguintes perguntas.

1 - Como funciona uma estrutura de controle de fluxo com `if`, `else` e `elif`? Escreva com suas próprias palavas como funciona cada uma dessas estruturas.

2 - Essa fun����o possui diversas opera����es, s��o elas `<=`, `%`, `*`, `==`, `+=` e `or`. Escreva um pequeno texto explicando o funcionamento de cada uma delas, em quais tipos de va

> Exemplo: O operador `and` em python é um operador lógico que realiza uma opera����o entre dois valores boolenos (`True` ou `False`):`bool1 and bool2`, que tem como saída outro valor

3 - Imagine que você esteja ensinando uma outra pessoa a realizar a mesma tarefa que essa fun����o em python est�� executando, imagine que você precisa entregar a essa pessoa uma folha
~~~

daniel_21_03_2024_baskara.py

```
FisComp.py/notas_de_aula/notas_2024_1/codigos_alunos/daniel_21_03_2024_baskara.py
# Esse código foi desenvolvido pelo aluno Daniel
# durante uma aula de computação básica para física
# no dia 21/03/2024

import math

a = float(input("Digite o valor a: "))
b = float(input("Digite o valor b: "))
c = float(input("Digite o valor c: "))

x = ((-b + math.sqrt(math.pow(b,2.0)-4.0*a*c))/(2*a))
xx = ((-b - math.sqrt(math.pow(b,2.0)-4.0*a*c))/(2*a))

print("A primeira raiz é : "+"{:10.4f}".format(x))
print("A segunda raiz é : "+"{:10.4f}".format(xx))
```
~~~