Lógica de Programação

Funções

Exercício

1) Faça um algoritmo que mostre o tipo de 2 variáveis distintas
2) Faça um algoritmo que gere um número aleatório entre 5 e 50
3) Faça um algoritmo coloca todo o texto de uma string em maiúsculo
4) Faça um algoritmo coloca todo o texto de uma string em minúsculo
5) Faça um algoritmo que substitua uma letra de uma string

Introdução

Funções são um dos conceitos mais importantes em programação. Elas nos permitem **encapsular** um bloco de código que realiza uma tarefa **específica**, para que possamos reutilizá-lo em diferentes partes do nosso programa.

Definindo Funções

Em Python, definimos uma função usando a palavra-chave def, seguida pelo nome da função, parênteses ( ) que podem incluir parâmetros e dois pontos :. O código da função é indentado em relação à definição da função.

```
def minha_funcao():
    print("Olá, mundo!")
```

Chamando Funções

Depois de definir uma função, podemos “**chamá-la**” ou “**invocá-la**” usando seu nome seguido por parênteses.

```
def minha_funcao():
    print("Olá, mundo!")

minha_funcao()
```

Parâmetros e Argumentos

Podemos passar informações para uma função incluindo parâmetros na definição da função. Quando chamamos a função, fornecemos os valores para esses parâmetros, que são chamados de argumentos.

```
def saudacao(nome):
    print(f"Olá, {nome}!")

saudacao("Maria")
```

Valores de Retorno

Funções podem retornar valores que podem ser usados em outras partes do nosso programa. Usamos a palavra-chave return para isso.

```
def quadrado(numero):
    return numero ** 2


x = quadrado(3)
print(x)
```

Data e Tempo

O Python oferece ferramentas robustas para manipulação de datas e horários, tornando-o ideal para diversas aplicações, desde cálculos simples até análises complexas de tempo. Nesta aula completa, exploraremos as principais funcionalidades do módulo datetime e aprenderemos a realizar diversas operações com datas e horários em Python.

Datetime

O módulo datetime é a principal biblioteca para trabalhar com datas e horários em Python. Ele fornece diversos objetos e funções para criar, formatar, comparar e manipular datas e horários de forma precisa e eficiente.

```python
from datetime import date

data_atual = date.today()
print(data_atual)
```

Usando datetime.time

```python
from datetime import time

horario_agora = time(hour=15, minute=30, second=55)
print(horario_agora)


horario_reuniao = time(hour=10, minute=0)
print(horario_reuniao)
```

Usando datetime.datetime

```python
from datetime import datetime


data_hora_agora = datetime.now()
print(data_hora_agora)

# Criando um datetime específico
data_hora_evento = datetime(year=2023, month=12, day=25, hour=20, minute=30)
print(data_hora_evento)
```

Acessando Atributos de Datas e Horários

```python
from datetime import datetime

data_hora = datetime.now()

# Acessando ano, mês, dia, hora, minuto e segundo
ano = data_hora.year
mes = data_hora.month
dia = data_hora.day
hora = data_hora.hour
minuto = data_hora.minute
segundo = data_hora.second

print(f"Ano: {ano}")
print(f"Mês: {mes}")
print(f"Dia: {dia}")
print(f"Hora: {hora}")
print(f"Minuto: {minuto}")
print(f"Segundo: {segundo}")
```

Formatando Datas e Horários

```python
from datetime import datetime

data_hora = datetime.now()

# Formatando data no estilo dd/mm/aaaa
data_formatada = data_hora.strftime("%d/%m/%Y")
print(f"Data formatada: {data_formatada}")

# Formatando hora no estilo hh:mm:ss
hora_formatada = data_hora.strftime("%H:%M:%S")
print(f"Hora formatada: {hora_formatada}")

# Formatando data e hora completas
data_hora_formatada = data_hora.strftime("%d/%m/%Y %H:%M:%S")
print(f"Data e hora formatadas: {data_hora_formatada}")
```

Comparação de Datas e Horários

```python
from datetime import date, datetime

data_1 = date(year=2024, month=5, day=7)
data_2 = date(year=2024, month=5, day=8)

# Comparando se data_1 é anterior a data_2
if data_1 < data_2:
    print("data_1 é anterior a data_2")
else:
    print("data_1 não é anterior a data_2")
```

Extraindo Datas de Strings

```python
from datetime import datetime

data_string = "07/05/2024"

# Convertendo string para objeto datetime
data_objeto = datetime.strptime(data_string, "%d/%m/%Y")
print(f"Data objeto: {data_objeto}")
```

Módulos Adicionais para Datas e Horários

- calendar: Funções para trabalhar com calendários, como obter o número de dias em um mês.
- time: Funções para medir tempo, como calcular o tempo de execução de um código.
- dateutil: Biblioteca com funcionalidades avançadas para manipulação de datas e horários.

Exercícios

Faça um algoritmo que compare se um ano é ou não anterior ao outro

Faça um algoritmo que mostre a data de hoje formatada, ex: (07/05/2024)

Matemática com Python

Python possui um conjunto de funções matemáticas integradas, incluindo um extenso módulo matemático, que permite realizar tarefas matemáticas com números.

Funções matemáticas integradas

As funções min( ) e max( ) podem ser usadas para encontrar o valor mais baixo ou mais alto em um iterável:

```
x = min(5, 10, 25)
y = max(5, 10, 25)

print(x)
print(y)
```

Funções matemáticas integradas

A função retorna o valor de x elevado à potência de y (x y).pow(x, y)

```
x = pow(4, 3)

print(x)
```

Exercício

Faça um conversor de moedas que converta dólar em real

Manipulação de Arquivos

A manipulação de arquivos é uma habilidade essencial para qualquer programador. Com o Python, você pode realizar diversas operações em arquivos de texto, como leitura, escrita, atualização e organização. Esta aula irá te guiar pelos conceitos básicos e te mostrar como usar as ferramentas do Python para trabalhar com arquivos de forma eficiente.

Abrindo e Fechando Arquivos

Antes de realizar qualquer operação, é necessário abrir o arquivo desejado. Isso é feito utilizando a função open( ), que recebe o nome do arquivo e o modo de abertura como parâmetros. Os modos de abertura mais comuns são:

Abrindo e Fechando Arquivos

- **r:** Abre o arquivo para leitura (padrão)
- **w:** Abre o arquivo para escrita (sobrescreve o conteúdo existente)
- **a:** Abre o arquivo para anexação (adiciona novo conteúdo ao final do arquivo)
- **r+:** Abre o arquivo para leitura e escrita
- **w+:** Abre o arquivo para escrita e leitura (sobrescreve o conteúdo existente)

Abrindo e Fechando Arquivos

```
# Abrindo um arquivo para leitura
arquivo = open('meu_arquivo.txt', 'r')

# Abrindo um arquivo para escrita
arquivo = open('meu_arquivo.txt', 'w')

# Abrindo um arquivo para anexação
arquivo = open('meu_arquivo.txt', 'a')
```

Abrindo e Fechando Arquivos

Após abrir o arquivo, é importante fechá-lo quando terminar de usá-lo. Isso garante que as alterações sejam salvas e libera os recursos do sistema.

```
# Fechando o arquivo
arquivo.close()
```

Lendo Conteúdo de Arquivos

Existem diversas maneiras de ler o conteúdo de um arquivo. Uma maneira comum é usar a função read( ), que retorna todo o conteúdo do arquivo como uma string.

```
arquivo = open('meu_arquivo.txt', 'r')
conteudo = arquivo.read()
arquivo.close()

print(conteudo)
```

Escrevendo em Arquivos

Para escrever no conteúdo de um arquivo, você pode usar a função write( ), que recebe a string que deseja escrever como parâmetro.

```
# Escrevendo no arquivo
arquivo = open('meu_arquivo.txt', 'w')
arquivo.write('Esta é uma nova linha no arquivo.\n')
arquivo.close()
```

Atualizando Conteúdo de Arquivos

Se você deseja atualizar o conteúdo de um arquivo existente, pode utilizar a função seek( ), que permite mover o ponteiro de leitura para uma posição específica no arquivo, e a função write( ), para escrever novo conteúdo.

```
# Atualizando uma linha específica do arquivo
arquivo = open('meu_arquivo.txt', 'r+')
arquivo.seek(30)  # Move o ponteiro para a posição 30
arquivo.write('Novo texto para substituir a linha antiga.\n')
arquivo.close()
```

Excluindo Arquivos

Para excluir um arquivo, você pode usar a função os.remove( ), que recebe o nome do arquivo como parâmetro.

```
import os

# Excluindo um arquivo
os.remove('arquivos/meu_arquivo.txt')
```

Pacotes Úteis

- **csv:** Para ler e escrever arquivos CSV.
- **json:** Para ler e escrever arquivos JSON.
- **shutil:** Para copiar, mover e excluir arquivos e diretórios de forma mais robusta.
- **pathlib:** Uma interface orientada a objetos para trabalhar com caminhos de arquivos e diretórios.

Por hoje é só...

Hoje vimos muita coisa, vamos descansar que amanhã tem mais!