Aula 12

Resumo

Abaixo temos um exemplo completo do que vimos até o momento das classes do Python.
Módulo funcoes.py

```python
"""módulo funcoes.py com os decoradores do programa"""
from time import sleep, time


def decorador_timer(func):
    """função decoradora para controlar o tempo de execução de uma função"""
    def interna(*args, **kwargs):
        tempo: int = 2
        print(f'Iniciando o timer em {tempo} segundos.')
        sleep(tempo)
        inicio = time()
        resultado = func(*args, **kwargs)
        fim = time()
        print('Finalizando o timer.')
        print(f'O tempo de execução foi de {fim - inicio} segundos.')
        return resultado
    return interna
```

Módulo classes.py

```python
"""classes.py"""
import random
from funcoes import decorador_timer


class Base:
    """classe base para servir de referência para as outras classes"""

    def __init__(self, nome: str, vida: int) -> None:
        self.nome: str = nome
        self.vida: int = vida
        self.dano: int = 0
        self.expr: int = 0

    def ataca(self, alvo: 'Base'):
        """método para atacar algum alvo"""
        alvo.vida -= self.dano
        print(f'{self.nome} atacou {alvo.nome} com {self.dano} de dano.')

    def __str__(self) -> str:
        return f'Nome: {self.nome} | Vida: {self.vida}'
```

```python
class Monstro(Base):
    """classe monstro que é capaz de gerar uma lista de inimigos
    também vai herdar a classe Base"""
    lista_nomes: list[str] = ['Goblin', 'Orc', 'Troll', 'Dragão']
    lista_monstros: list['Monstro'] = []

    def __init__(self, nome: str, vida: int, dano: int, xp: int) -> None:
        super().__init__(nome, vida)
        self.dano: int = dano
        self.expr: int = xp

    @classmethod
    def gera_monstro(cls, novo_nome: str) -> 'Monstro':
        """método gera_monstro usado para criar uma instância da classe
        Monstro"""
        nome: str = novo_nome
        vida: int = random.randint(1, 100)
        dano: int = random.randint(1, 10)
        experiencia: int = random.randint(1, 10)
        return cls(nome, vida, dano, experiencia)

    @decorador_timer
    @staticmethod
    def fabrica_monstro() -> None:
        """método fabrica_monstro usado para criar uma lista de monstros"""
        for _ in range(10):
            nome: str = random.choice(Monstro.lista_nomes)
            Monstro.lista_monstros.append(Monstro.gera_monstro(nome))

    def __str__(self) -> str:
        return f'Nome: {self.nome} | Vida: {self.vida} | Dano: {self.dano}'
```

Módulo main.py

```python
"""main.py"""

from classes import Monstro

if __name__ == '__main__':
    Monstro.fabrica_monstro()
    for monstro in Monstro.lista_monstros:
        print(monstro)
```

Módulo os

O módulo os do Python é uma biblioteca integrada que fornece uma interface para interagir com o sistema operacional. Ele oferece uma ampla gama de funções para trabalhar com arquivos, diretórios, caminhos de arquivo, variáveis de ambiente e muito mais.

Através do módulo os, é possível realizar várias operações comuns relacionadas ao sistema operacional, independentemente da plataforma em que o código Python esteja sendo executado. Ele é particularmente útil quando você precisa escrever código que seja executado em diferentes sistemas operacionais, como Windows, macOS e Linux.

Aqui estão algumas das principais funcionalidades oferecidas pelo módulo os:

1. Manipulação de diretórios e arquivos:
    a. Criar, renomear e excluir diretórios.

   b. Listar o conteúdo de um diretório.
   c. Verificar a existência de um arquivo ou diretório.
   d. Obter informações sobre um arquivo, como tamanho e data de modificação.
   e. Manipular caminhos de arquivo para trabalhar com diferentes formatos.
2. Executar comandos do sistema operacional:
   a. Executar comandos do shell ou linha de comando.
   b. Capturar a saída e o código de retorno desses comandos.
3. Gerenciamento de processos:
   a. Iniciar um novo processo.
   b. Controlar e comunicar-se com processos em execução.
   c. Obter informações sobre o processo atual.
4. Variáveis de ambiente:
   a. Acessar variáveis de ambiente do sistema operacional.
   b. Definir e modificar variáveis de ambiente.

Essas são apenas algumas das funcionalidades mais comuns do módulo os. Para utilizar essas funcionalidades, você precisa importar o módulo no seu código Python. Por exemplo:

```python
import os

# Exemplos de uso
os.mkdir('diretorio')  # Cria um diretório
os.listdir('.')  # Lista o conteúdo do diretório atual
os.system('ls -l')  # Executa o comando 'ls -l' no shell
```

O módulo os é muito poderoso e flexível, permitindo que você crie aplicativos Python que interajam com o sistema operacional de maneira eficiente e portável. Para obter mais detalhes sobre as funcionalidades oferecidas pelo módulo os, você pode consultar a documentação oficial do Python.

https://docs.python.org/pt-br/3/library/os.html

Módulo time

O módulo time é uma biblioteca padrão do Python que fornece várias funções relacionadas à manipulação do tempo. Ele permite que você trabalhe com informações de data, tempo e medição de tempo decorrido.

Aqui estão algumas das principais funções e recursos oferecidos pelo módulo time:

- time(): Retorna o tempo atual em segundos desde a época (1º de janeiro de 1970). É útil para medir a duração de um processo ou calcular a diferença entre dois momentos no tempo.

```python
import time

agora = time.time()
print(agora)
```

- sleep(secs): Pausa a execução do programa por um determinado número de segundos (secs). É útil para adicionar atrasos ou pausas em seu código.

```python
import time

print("Começando ... ")
time.sleep(2)  # Pausa por 2 segundos
print("Continuando após a pausa.")
```

Essas são apenas algumas das funções oferecidas pelo módulo time. Existem outros recursos disponíveis, como formatação personalizada de datas e tempos, manipulação de tempo em termos de anos, meses, dias, horas, etc., e muito mais.

https://docs.python.org/pt-br/3/library/time.html

Exercícios para Praticar

1. Verifique se um diretório existe.
2. Crie um novo diretório.
3. Renomeie um arquivo.
4. Liste todos os arquivos de um diretório.
5. Liste apenas os diretórios de um diretório.
6. Liste todos os arquivos com uma determinada extensão em um diretório.
7. Verifique se um arquivo existe.
8. Verifique se um caminho é um arquivo ou diretório.
9. Obtenha o tamanho de um arquivo em bytes.
10. Obtenha a data de modificação de um arquivo.
11. Delete um arquivo.
12. Delete um diretório vazio.
13. Delete um diretório e todos os seus arquivos e subdiretórios.
14. Copie um arquivo de um local para outro.
15. Mova um arquivo de um local para outro.
16. Execute um comando do sistema operacional.
17. Capture a saída de um comando do sistema operacional.
18. Acesse a variável de ambiente PATH.
19. Defina uma nova variável de ambiente.
20. Liste todas as variáveis de ambiente disponíveis.

21. Escreva um programa que exiba o tempo atual em segundos desde a época.
22. Crie uma função que pausa a execução do programa por 5 segundos e depois exiba uma mensagem.
23. Escreva um programa que converta um tempo em segundos desde a época em uma string de data e hora legível para humanos.
24. Crie uma função que exiba a data e hora atual no formato "AAAA-MM-DD HH:MM:SS".
25. Escreva um programa que calcule a diferença de tempo entre duas datas fornecidas pelo usuário.
26. Crie uma função que exiba a hora atual no formato de 12 horas, indicando se é AM ou PM.
27. Escreva um programa que verifique se um ano fornecido pelo usuário é bissexto ou não.
28. Crie uma função que exiba a data e hora atual em diferentes fusos horários, como UTC, GMT-3, etc.
29. Escreva um programa que calcule a idade em anos, meses e dias com base em uma data de nascimento fornecida pelo usuário.
30. Crie uma função que exiba o nome do dia da semana correspondente a uma data fornecida pelo usuário.
31. Escreva um programa que exiba a quantidade de dias restantes até o próximo Ano Novo.
32. Crie uma função que exiba a hora atual com uma precisão de milissegundos.
33. Escreva um programa que converta uma quantidade de segundos em horas, minutos e segundos.
34. Crie uma função que exiba a data e hora atual, arredondada para o minuto mais próximo.
35. Escreva um programa que calcule o tempo decorrido entre a data e hora atual e uma data e hora fornecidas pelo usuário.
36. Crie uma função que exiba a data atual no formato "DD/MM/AAAA".
37. Escreva um programa que verifique se uma data fornecida pelo usuário é válida ou não.
38. Crie uma função que exiba a quantidade de segundos desde o início do dia atual.
39. Escreva um programa que exiba a data e hora atuais, formatadas de acordo com as preferências do usuário.
40. Crie uma função que exiba o número total de dias em um determinado mês e ano fornecidos pelo usuário.