Aula 4

Linguagem de Programação Aplicada

Prof. Renan Portela Jorge

1

Conversa Inicial

2

O que estudaremos nessa aula?

- **Nessa aula veremos**
  - **Como utilizar Interrupções**
  - **Entender e utilizar Decorators**
  - **Continuaremos nossos estudos com Padrões de Design (*Design Pattern*)**

3

Eventos

4

Interrupções de teclado

- **Para fazer interrupções de teclado em Python, utiliza-se a biblioteca keyboard**
- **Esta requer permissões de acesso ao teclado, o que pode variar dependendo do SO**

5

Interrupções de tempo

- **Para criar eventos de timer em Python sem bloquear a execução do programa, uma alternativa é usar a biblioteca sched**
- **Fornece recursos para agendar e executar funções em momentos específicos ou após determinados intervalos de tempo**

6

Decorators

7

- São utilizados para modificar ou estender a funcionalidade de funções ou classes sem precisar alterar seu código interno
- Os decorators são implementados usando a sintaxe do @
- Existem dois tipos principais de decorators em Python
  - Decorators de função
  - Decorators de classe

8

Decorators de função

- São aplicados a funções individuais e permitem adicionar funcionalidades extras antes, depois ou ao redor da função decorada

9

Decorators de classe

- Os decorators de classe são aplicados a classes inteiras e permitem modificar ou estender o comportamento da classe

10

Compreensão de Lista (*List Comprehension*)

11

- Permite criar uma nova lista a partir de uma expressão ou iteração em uma única linha
  - Caso com somente uma condição

```
lista_pares_quadrado = []
for num in range(10):
    if num % 2 == 0:
        lista_pares_quadrado.append(num*num)
```

crie uma nova lista com esses elementos que dentro do intervalo obedecem a seguinte condição

```
lista_pares_quadrado = [ num*num for num in range(10) if num % 2 == 0]
```

12

- **Caso com somente duas condições**

```
lista_pares_quadrado = []
for num in range(10):
    if num % 2 == 0:
        lista_pares_quadrado.append(num*num)
    else:
        lista_pares_quadrado.append('impar')
```

```
lista_pares_quadrado = [ num*num if num % 2 == 0 else 'impar' for num in range(10)]
```

13

Padrão de Design de Comportamental
Mediador (*Mediator*)

14

Design Pattern Mediator

- **Útil quando você tem vários componentes que precisam se comunicar entre si de forma desacoplada**
- **Gerencia as interações entre os componentes, promovendo a baixa dependência**

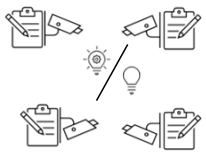
15

Padrão de Design de Criacional
Observador

16

Design Pattern*: *Observer

- **Permite que objetos observadores sejam notificados e atualizados quando ocorrem mudanças no objeto observado**
- **Isso permite um acoplamento flexível e desacoplado, facilitando a comunicação e a reatividade em um sistema**



17