FERNANDO NISHIO

POO EM PYTHON

feat Felipe Trentin.

Press any key to continue...

POO EM PYTHON

>O que é POO?
>Conceitos
>Classes, objetos e atributos
>Como criar métodos
>Encapsulamento
>Herança
>Polimorfismo

Press any key to continue...

O QUE É POO

>POO ou programação orientada a objetos, trata-se de um paradigma de programação, na qual o código é organizado em ”objetos”. Esses objetos representam entidades do mundo real ou conceitos e combinam dados e comportamentos.
>A POO tem como intuito, modelar entidades no ambiente real, usando as interações entre eles para resolver problemas.

Press any key to continue...

CONCEITOS

- Classe
- Objeto
- Atributo
- Método
- Encapsulamento
- Abstração
- Herança
- Polimorfismo

Press any key to continue...

CLASSES, OBJETOS E ATRIBUTOS

>Classe: Um molde ou template para criar objetos. Define as propriedades (atributos) e métodos (funções) que os objetos daquela classe terão.

>Atributos:São definidos dentro de uma classe e representam as características ou o estado de um objeto.

>Objetos:São uma instância de classe, usadas para representar entidades no mundo real. Elas contém características (atributos) que podem ser acessadas

Press any key to continue...

CLASSES, OBJETOS E ATRIBUTOS

class Pokemon: Untitled-1

```python
class Pokemon:
    # Inicializador que recebe o valor do objeto vindo de fora da classe
    def __init__(self,nome,tipo,nivel):
        self.nome = nome #Atributo para guardar o nome
        self.tipo = tipo #Atributo para guardar o tipo
        self.nivel = nivel #Atributo para guardar o nível

Pikachu = Pokemon("Ronaldo","Elétrico", 20) #Criação de um obejto (instância) para a classe Pokemon
```


Press any key to continue...

COMO CRIAR MÉTODOS

>Métodos são funcões (ações) atribuídas a uma classe, usadas para modificar dados e representar os comportametos de um objeto.

Press any key to continue...

COMO CRIAR MÉTODOS

class Pokemon:.py > ...

```python
class Pokemon:
    # Inicializador que recebe o valor do objeto vindo de fora da classe
    def __init__(self,nome,tipo,nivel):
        self.nome = nome #Atributo para guardar o nome
        self.tipo = tipo #Atributo para guardar o tipo
        self.nivel = nivel #Atributo para guardar o nível
    def mostrar_atributos(self):#Criando o método
        print(f"Nome: {self.nome}\n Tipo: {self.tipo}\n Nível: {self.nivel}")#Instruções do método

Pikachu = Pokemon("Ronaldo","Elétrico", 20) #Criação de um obejto (instância) para a classe Pokemon

Pikachu.mostrar_atributos()#Chamando o método

```

```
Nome: Ronaldo
 Tipo: Elétrico
 Nível: 20
```

Press any key to continue...

COMO CRIAR MÉTODOS

raio.py > ...

```python
# Método que retorna um valor baseado no estado do objeto
class Circulo:
    pi = 3.14159  # Atributo de classe

    def __init__(self, raio):
        self.raio = raio

    def area(self):
        return self.pi * (self.raio ** 2)

# Uso do método
circulo = Circulo(5)
print(f"A área do círculo é: {circulo.area():.2f}")
```


Press any key to continue...

ENCAPSULAMENTO

>O encapsulamento trata-se de uma técnica de POO cujo propósito é restringir o acesso e modificação de atributos de um objeto.

>Evita alterações acidentais ou maliciosas de terceiros

>Encapsulamento público

>Encapsulamento privado

Press any key to continue...

ENCAPSULAMENTO

```python
class Pokemon:
    #Inicializador que recebe o valor vindo de fora da classe
    def __init__(self,nome,tipo,nivel):
        self.nome = nome #Atributo para guardar o nome
        self.tipo = tipo #Atributo para guardar o tipo
        self.__nivel = nivel #Atributo para guardar o nível, note que ele está protegido
    def adc_nível(self,valor): # criação de um método para modificar o atributo nível
        self.__nivel += valor
    def mostrar_atributos(self):
        print(f"Nome : {self.nome}\n Tipo : {self.tipo}\n Nível : {self.__nivel}")

Pikachu = Pokemon("Ronaldo","Elétrico",20) #Criação de um objeto para a classe Pokemon

x = int(input("Seu pokemon recebeu exp! Qunatos níveis ele aumentou?"))
Pikachu.adc_nível(x) #Só é possível modificar nível quando se usa a classe
Pikachu.mostrar_atributos()

```

```
Press any key to continue...
```

HERANÇA

>Uma das grandes vantagens de POO é a herança, ela permite a reutilização de código de uma maneira simples.

>Ela possibilita que classes (subclasses) herdem (copiem) características de uma classe maior (Superclasses)

Press any key to continue...

HERANÇA

```python
class Pokemon:
    #Inicializador que recebe o valor vindo de fora da classe
    def __init__(self,nome,tipo,nivel):
        self.nome = nome #Atributo para guardar o nome
        self.tipo = tipo #Atributo para guardar o tipo
        self.__nivel = nivel #Atributo para guardar o nível, note que ele está protegido
    def adc_nível(self,valor): # criação de um método para modificar o atributo nível
        self.__nivel += valor
    def mostrar_atributos(self):
        print(f"Nome : {self.nome}\n Tipo : {self.tipo}\n Nível : {self.__nivel}")

class Pikachus (Pokemon):
    def __init__(self,nome,tipo,nivel,shiny):
        super().__init__(nome,tipo,nivel)
        self.shiny = shiny
        if shiny == random.randint(1,8192):
            self.shiny = 1
        else:
            self.shiny =0
    def choque(self):
        print(f"Pikachu usou choque do trovão! \n")

y = random.randint(1,8192)

Pikachu = Pikachus("Ronaldo","Elétrico",20,y) #Criação de um objeto para a classe Pokemon

if Pikachu.shiny == 1:
    print("**** é um shiny**** \n")

Pikachu.choque()

x = int(input("Seu pokemon recebeu exp! Quantos níveis ele aumentou? \n"))

Pikachu.adc_nível(x) #Só é possível modificar nível quando se usa a classe

Pikachu.mostrar_atributos()
```

```
Press any key to continue...
```

POLIMORFISMO

>O polimorfismo permite que uma função ou método compartilhe o mesmo nome em várias classes, mas execute diferentes implementações.

>Essa flexibilidade é usada para que diferentes tipos de objetos possam responder ao mesmo comando de acordo com suas características.

Press any key to continue...

POLIMORFISMO

```python
class Pokemon:
    def __init__(self,nome,tipo,nivel):
        self.tipo = tipo #Atributo para guardar o tipo
        self.__nivel = nivel #Atributo para guardar o nível, note que ele está protegido
    def adc_nível(self,valor): # criação de um método para modificar o atributo nível
        self.__nivel += valor
    def mostrar_atributos(self):
        print(f"Nome : {self.nome}\n Tipo : {self.tipo}\n Nível : {self.__nivel}")
    def ataque(self):
        pass

class Pikachus (Pokemon):
    def __init__(self,nome,tipo,nivel,shiny):
        super().__init__(nome,tipo,nivel)
        self.shiny = shiny
        if shiny == random.randint(1,8192):
            self.shiny = 1
        else:
            self.shiny =0
    def ataque(self):
        print(f"{self.nome} usa choque do trovão!")

class Charizards (Pokemon):
    def __init__(self,nome,tipo,nivel,shiny):
        super().__init__(nome,tipo,nivel)
        self.shiny = shiny
        if shiny == random.randint(1,8192):
            self.shiny = 1
        else:
            self.shiny =0
    def ataque(self):
        print(f"{self.nome} usou lança-chamas")
```

Press any key to continue...

POLIMORFISMO

```python
y = random.randint(1,8192)
z = random.randint(1,8192)

Pikachu = Pikachus("Ronaldo","Elétrico",20,y) #Criação de um objeto para a classe Pokemon
Charizard =Charizards("Jonas","Fogo",36,z)
if Pikachu.shiny == 1:
    print("**** é um shiny**** \n")
if Charizard.shiny == 1:
    print("**** é um shiny**** \n")

Pikachu.ataque()
Charizard.ataque()
x = int(input("Seu Pikachu ganhou exp! quantos níveis ele ganhou?"))
Pikachu.adc_nível(x)
Pikachu.mostrar_atributos()
Charizard.mostrar_atributos()
```

```
Press any key to continue...
```

EXERCÍCIOS

Press any key to continue...

EXERCÍCIOS

>1.Criar a classe FiguraGeometrica com os métodos calcular perímetro e calvular área. Implementar para circunferência, retângulo e triângulo baseado nessa classe mãe

>2. Crie um sistema para gerenciar funcionários de uma empresa.Crie uma classe Funcionario com os atributos nome, cargo, salario e horas_trabalhadas (inicialmente, horas_trabalhadas deve ser 0).Adicione um método registrar_horas que aumenta as horas trabalhadas pelo funcionário.Adicione um método calcular_pagamento que calcula o pagamento do funcionário com base no salário e nas horas trabalhadas.

>Adicione uma classe Empresa que gerencia uma lista de funcionários.Na classe Empresa, adicione métodos para:adicionar_funcionario: adicionar um novo funcionário à lista de funcionários.processar_pagamentos: calcular o pagamento de todos os funcionários, mostrar os detalhes de cada pagamento e zerar as horas trabalhadas após o cálculo.

Press any key to continue...

EXERCÍCIOS

>3.Imagine que você está desenvolvendo um sistema para controlar robôs que se movem em um grid bidimensional. O robô recebe instruções de movimento e precisa navegar pelo grid, evitando obstáculos.
>Crie uma classe `Robô` com os atributos: `x` e `y`: posição atual do robô no grid (começando em `(0, 0)` por padrão). `direção`: direção para onde o robô está voltado, começando para o norte (`N`).
>A classe `Robô` deve ter métodos para: `mover()`: move o robô para frente na direção em que ele está voltado. Cada movimento avança uma célula no grid. `virar_esquerda()`: altera a direção do robô para a esquerda (Norte → Oeste → Sul → Leste → Norte). `virar_direita()`: altera a direção do robô para a direita (Norte → Leste → Sul → Oeste → Norte).
>Crie uma classe `Grid` para representar o espaço onde o robô se move:A classe deve ter uma lista de obstáculos, representados como coordenadas (exemplo: `[(1, 1), (2, 2)]`).Deve ter métodos para:`adicionar_obstaculo(x, y)`: adiciona um obstáculo no grid.`existe_obstaculo(x, y)`: retorna `True` se houver um obstáculo na posição `(x, y)`, caso contrário `False`.
>Implemente um método `executar_instrucoes` que recebe uma lista de instruções (exemplo: `["mover", "virar_direita", "mover"]`) e faz o robô executá-las, parando caso o movimento seja bloqueado por um obstáculo. Se o robô tentar se mover para uma célula com obstáculo, ele deve parar e imprimir uma mensagem de erro informando a posição do obstáculo.Adicione um método `posicao_atual` na classe `Robo` para imprimir a posição `(x, y)` e a direção atual.

Press any key to continue...

FIM, OBRIGADO POR COMPARECEREM :)