Maria Clara Vilas Boas

\# Cientista de dados @ iFood

\# Futura cientista da computação

\# Mãe de gatos

POO: Pythonices Orientadas a Objetos

Programação **O**rientada a **O**bjetos em **Python**

Motivação

Pessoas que estão iniciando seus estudos nesse paradigma de programação, em python, e que já tiveram experiências com outras linguagens, ou não.

O QUE É POO?

Programação Orientada a Objetos é um paradigma de programação, e consiste numa técnica de desenvolvimento de software que busca imitar o mundo real;

Assim os programas se tornam mais reutilizáveis, confiáveis e inteligíveis;

Pilares de POO

Um programa desenvolvido de acordo com a POO incorpora alguns princípios importantes:

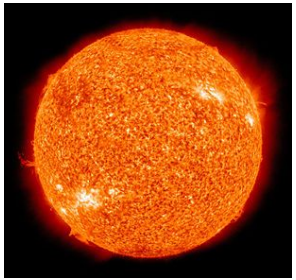
- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Abstração

Capacidade e habilidade em ater-se aos aspectos essenciais em um contexto.

-> Isolar o que importa e desprezar as características

menos importantes.



Classes

Uma classe é uma **abstração** de uma **entidade existente** no **domínio de um problema** que está sendo modelado através de um sistema computacional.

É usada para **criar um objeto**.

Cada objeto criado a partir de uma mesma classe terá características semelhantes ou mesmo idênticas.

ATRIBUTOS E MÉTODOS

ATRIBUTOS Incorporam todas as características de um conjunto de objetos específicos.

Também é incorporada à classe as funções que operam sobre

seus atributos, que são os **MÉTODOS**.

```python
class Gato():

    def __init__(self, nome, idade_em_anos):

        print('Inicializando a classe Gato')

        self.__idade_filhote = 1
        self.nome = nome
        self.idade_em_anos = idade_em_anos

    def gato_eh_filhote(self) -> bool:
        return self.idade_em_anos <= self.__idade_filhote

if __name__ == "__main__":
    gatito = Gato(nome='Leônidas', idade_em_anos=0.3)

    print(gatito.nome)
    print(gatito.idade_em_anos)
    print(gatito.gato_eh_filhote())
```

Definição da classe

Inicializador da classe
(método “construtor”)

Método da classe

Objeto = Instância da classe

```
Inicializando a classe Gato
Leônidas
0.3
True
```

Método 'construtor' __init__ , e sobrecarga

```python
class Gato():

    def __init__(self, nome, idade_em_anos, cor_olho=None):

        print('Inicializando a classe Gato')

        self.__idade_filhote = 1
        self.nome = nome
        self.idade_em_anos = idade_em_anos
        self.cor_olho = cor_olho
```

Python

Java

```java
public class calculadora{

  private String modelo;
  private String marca;
  private String uso;

  //Sobrecarga de construtores.
  public calculadora(){
    // esse é o construtor padrão que o programa cria para todas as classes.
  }
  public calculadora(String marca,String modelo){
           this.marca=marca;
           this.modelo=modelo;
  }

  public calculadora(String marca,String modelo,String uso){
           this.marca=marca;
           this.modelo=modelo;
           this.uso=uso;
  }
```

Código JAVA retirado na página: https://www.devmedia.com.br/

Encapsulamento

Restringir a visibilidade dos atributos de um objeto.

Transforma os atributos em uma caixa preta, onde somente os métodos do objeto terão acesso.

Vantagens:

- Melhora a Legibilidade do Código
- Facilita a Manutenção
- Favorece a Reutilização

Encapsulamento em JAVA

```java
public class TV {
  private int tamanho;
  private int canal;
  private int volume;
  private boolean ligada;
  public TV(int tamanho) {
    this.tamanho = tamanho;
    this.canal = 0;
    this.volume = 0;
    this.ligada = false;
  }
  // abaixo vem todos os métodos construtores
```

Código retirado na página: https://www.devmedia.com.br/

Encapsulamento em Python

```python
class Gato():

    def __init__(self, nome, idade_em_anos, cor_olho=None):

        print('Inicializando a classe Gato')

        self.__idade_filhote = 1
        self.nome = nome
        self.idade_em_anos = idade_em_anos
        self.cor_olho = cor_olho
```

Herança

HERANÇA É o mecanismo em pelo qual uma classe permite herdar os recursos (atributos e métodos) de outra classe. **Terminologias:**

- **Superclasse**
- **Subclasse**
- **Reutilização**

```python
# Python code to demonstrate how parent constructors
# are called.

# parent class
class Person():

        # __init__ is known as the constructor
        def __init__(self, name, idnumber):
                self.name = name
                self.idnumber = idnumber
        def display(self):
                print(self.name)
                print(self.idnumber)

# child class
class Employee(Person):
        def __init__(self, name, idnumber, salary, post):
                self.salary = salary
                self.post = post

                # invoking the __init__ of the parent class
                Person.__init__(self, name, idnumber)

```

Polimorfismo

Polimorfismo refere-se à capacidade das linguagens de programação de POO para diferenciar entre entidades com o mesmo nome com eficiência.

Polimorfismo

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")

    def language(self):
        print("Hindi is the most widely spoken language of

class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

    def language(self):
        print("English is the primary language of USA.")


obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
```

Um
imenso
obrigada
a presença
de **vocês**!