# **Aula 2 Programação Orientada a Objeto**

Tudo que tem na natureza, o objetivo é digitalizar esses objetos

**Keywords**(Palavras Chaves):

* OOP/POO - Programação Orientada a Objetos.
* Regra de Negócio é aproximadamente igual à Problema
  + Problemas no brasil
  + Economia.
* Modelagem de dados
* Resumo: Extensão: Herança.
* Agnóstico, adaptável, substituível 🡨 Programa devem ser
* API (Software que se comunica com outros softwares)
* \_\_main\_\_
* Docstring == \_\_doc\_\_ (retorna a documentação da classe)
* \_\_init\_\_: “Construtor”, Inicializador.
* Instanciamento
* Metodologia Scrum
* Scrum máster (O cara que cobra o serviço)
* Daily (breve apresentação do que vai fazer hoje)
* Sprint (temporada): relatório semanal
* Black(linter)
* “” 🡪 Delimitador de Texto
* Recursividade 🡪 executar repetidas vezes
* Variável == parâmetro == container == Caixa Registradora
* Argumento == valores
* def func2(parametro1, parametro2)
* DRY – Don’t repeat yourself (não se repita/ Evite repetições)
* Escopo (Limite)
* Tudo que vai dentro do ( ) é argumento.
* Data sanitation
* IDE – Integrated Development Environment (Ambiente de Desenvolvimento integrado)

POO é um paradigma da programação



Chicken

**Frango**

hen

**---//--- ---//--- ---//---**

**Objetos:**

Objeto (Victor março 2022) “Tudo aquilo que é visível, tangenciável, perceptível, e as vezes ou muitas vezes, dependendo da sua área, abstratas.”

Exemplo:

Borracha

 Atributos

Abstração

* Cor
* Tamanho: largura, comprimento.
* Densidade.
* Fabricante.
* Marca.

Objeto

* Preço.

Classe

**Eruser (Apagar)**

**Tudo pode ser representado por objeto**

**Todo objeto tem atributos**

Garbage(Lixo)

Garbage collector (Aliviador de memória)

Regras do Python

PEP 8 🡪Proposta de melhoramento do phython

Todo final do código de Python tem de ter um “enter” no final. Linha vazia.

Paradigmas:

* Funcional, OOP, Procedural, Data Driven (Orientada a dados), Programação Orientada a Eventos
* UML

**Modelagem de dados**

|  |
| --- |
| **class Cachorro:**  **def \_\_init\_\_(Self):**  **puss**    **#------------------------------------------**  **class Cachorro:**  **def \_\_init\_\_(Self, nome, idade, porte, raca ):**  **Self.nome=nome#Self referencia a classe cachorro. Modela/cria Propriedades** |

**Métodos Especiais/**Mágicos

* DUNDER METHODS (Anderline no começo e no fim do nome)

Caracterizam propriedade da classe

Tipos de dados

Strings: (Textos)

Funções:

.find()

.upper()

.title()

.split()

.swapcase() Inverte o maiúsculo pelo minúsculo e etc.

Int: (inteiros)

1, 44, 18056

Float(Real)

3.1, 44.7

Dicionário: Objeto

Tupla/Lista

# **Aula 3**

Note:

* Ctrl + ENTER = EXECUTAR
* SHIFT + ENTER = NOVA LINHA
* ESC + X = APAGAR
* ESC + M = MARCDOW
  + #TITULO1 ## TITULO2 ### TITULO3

Link do drive apostila:  
<https://drive.google.com/drive/folders/1wOe9R6P63K96kvAAB1tzbK7PsBMipN3i>

Aula 5 - **Parse**

Arquivo com a extensão ‘.gitignore’ sobe para o github

Pasta ‘venv’ é um ambiente virtualizado para cada projeto

<https://pypi.org/> - subir um programa, muito mais difícil que o git hub

**Modelagem de dados**

Sabesp



json

csv

xml

doc

pat

jpg

mp3

Estoque.csv

1. – INSOMNIA (Requisitor): Requisição, verbos HTTP, Trafego de informações, headers
   1. - WORKSPACE:

Request collection (coleção de requisição)

1. – Verbos http

Tudo que vai volta (nem tudo – buraco negro)

GET

POST

PUT

DELETE

CRUD –

Create (Crie - POST)

Read (Leia - GET)

Update (Atualize - PUT)

Delete (Delete)

POSTMAN

...

NAVEGADOR

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status

REQUISIÇÕES HTTP

1. Respostas de informação (100-199),
2. Respostas de sucesso (200-299),
3. Redirecionamentos (300-399)
4. Erros do cliente (400-499)
5. Erros do servidor (500-599).
6. DECORATOR (decorador):

Função que retorna outra função, decorador

@

Manipuladores de comportamentos

1. Diferenças entre anotações

Criando funções

**Def nome\_da\_fucao()**

Explicitando (nomeando) a função

nome\_da\_funcao

Chamando Função:

nome\_da\_funcao()