Introdução a P.O.O com Python

github.com/ferrazGab
gabrielfap479@gmail.com

• Sintaxe básica python

- `input` - receber dados
- `print` - Imprime um dado
- `def` - define uma função
- `if...else` - condição

github.com/ferrazGab
gabrielfap479@gmail.com

• Paradigma procedural

```
def hello():
    print(f"Hello, world!")

def hi(name):
    print(f"Hi, {name}!")

hello()
hi("SomeNameExample")
```

```
"""
Paradigma (forma) de programação marcada basicamente por instruções e lógica, o que força o programador a repetir código e o limita no quesito a organização.

"""
```

github.com/ferrazGab
gabrielfap479@gmail.com

• Paradigma P.O.O. (Orientação a objetos)

- Abstração;
- Encapsulamento;
- Herança;
- Polimorfismo.

github.com/ferrazGab
gabrielfap479@gmail.com

• Abstração

- **Representação de um objeto real em um virtual**

```
Class Cachorro(): // Objeto

    cor = "marrom" // Atributo

    def latir(): // Método
        print (f"Au")
```

```
"""
Atributos são características pertinentes a um objeto

Métodos são as ações de um objeto, ou seja, o que ele pode fazer
"""
```


github.com/ferrazGab


gabrielfap479@gmail.com

- Encapsulamento

```
class Cachorro():

    __cor = "marrom"
    def latir():
        print (f"Au")

    def getCor():
        return __cor

Inst = Cachorro() //*1
Cor = Cachorro.getCor()
print (Cor) //Erro
```

```
"""
*1 - Instância

Isso irá retornar um
erro, pois o atributo
cor (com dois
underlines precendendo
o nome) é privado a
classe Cachorro
"""
```

github.com/ferrazGab
gabrielfap479@gmail.com

• Herança

```
class Cachorro():

    def latir():
        print (f"Au")

class ViraLata(Cachorro):
    cor = "caramelo"

class Pitbull(Cachorro):
    cor = "Marrom"

A = ViraLata()
B = Pitbull()

print (A.cor) //Resultado = Caramelo
print (B.cor) //Resultado = Marrom
A.latir() // Resultado = Au
B.latir() // Resultado = Au
```

```
"""
Um objeto pode herdar de outro todas as suas características, que se tornam funcionais dentro de sua estrutura
"""
```

github.com/ferrazGab
gabrielfap479@gmail.com

• Polimorfismo

```
class Cachorro():

    cor = "preto"

class ViraLata(Cachorro):
    cor = "caramelo"

class Pitbull(Cachorro):
    cor = "Marrom"

A = ViraLata()
B = Pitbull()

print (A.cor) //Resultado = Caramelo
print (B.cor) //Resultado = Marrom
```

```
"""
Com o polimorfismo,
podemos herdar
características de uma
classe mãe, porém,
dependendo da
necessidade, podemos
redefini-los.
"""
```

github.com/ferrazGab
gabrielfap479@gmail.com

Estrutura em python

```
class Pessoa():

    def __init__(self, nome, idade): // Método inicializador
        self.nome = nome
        self.idade = idade

    def falar():
        print (f"Olá, mundo! Meu nome é {self.name}
        e tenho {self.idade} anos!")

Nome = input("Digite seu nome: ")
Idade = input(int("Digite a sua idade: ")
Inst = Pessoa(Nome, Idade)
Inst.falar()
```



github.com/ferrazGab


gabrielfap479@gmail.com

Vamos a prática?

github.com/ferrazGab
gabrielfap479@gmail.com

Todo código estará disponível no repositório FTC no Github

github.com/ferrazGab
gabrielfap479@gmail.com