

# Linguagem de Programação Orientada a Objetos I

Python – Objetos

Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

# Python Orientado a Objetos

- ▶ Como dito antes, Python é uma linguagem multi-paradigmas
- ▶ Logo, é possível programar Python com orientação a objetos, com os conceitos de classes, objetos, atributos, métodos, herança, etc.
- ▶ Os conceitos de O.O. não serão vistos nesta disciplina, visto que já foram vistos em Linguagem de Programação Orientada a Objetos I

# Classe

- ▶ A criação de classes em Python se dá através da palavra-chave class

```
[>>> class Aluno:  
[...     pass  
[...  
[>>> Aluno  
<class '__main__.Aluno'>  
[>>> Aluno()  
<__main__.Aluno object at 0x102566358>  
[>>> al1 = Aluno()  
[>>> al1  
<__main__.Aluno object at 0x102566400>  
[>>> al2 = Aluno()  
[>>> al2  
<__main__.Aluno object at 0x102566438>  
[>>> ]
```

# Classe

```
>>> class Aluno:  
...     pass  
...  
>>> AlunosEstudiosos = Aluno  
>>> Aluno  
<class '__main__.Aluno'>  
>>> AlunosEstudiosos  
<class '__main__.Aluno'>  
>>> AlunosEstudiosos.__name__  
'Aluno'  
>>> █
```

# Atributos

```
>>> class Aluno:  
...     nome=''  
...     curso=''  
...     cgu=0  
...  
>>> class Aluno:  
...     nome=None  
...     curso=None  
...     cgu=None  
...  
>>> al1 = Aluno()  
>>> al1.nome = 'Tales'  
>>> al1.curso = 'Computacao'  
>>> al1.cgu = 1234  
>>> al1  
<__main__.Aluno object at 0x10a053518>  
>>> al1.nome  
'Tales'  
>>> al1.curso  
'Computacao'  
>>> al1.cgu  
1234
```

# Métodos

```
class Circulo:  
  
    raio = None  
  
    def calcula_area(self):  
        self.area = 3.14 * (self.raio ** 2)  
  
    def calcula_volume(self, altura):  
        if (not self.area):  
            self.calcula_area()  
        self.volume = self.area * altura
```

# Construtores

- ▶ Utiliza-se o método `__init__`
- ▶ Não podemos ter mais de um `__init__` na classe
- ▶ Podemos usar default values

```
class Circulo:  
  
    raio = None  
  
    def __init__(self, raio):  
        self.raio = raio  
  
c1 = Circulo(5)  
print("Raio de c1: ", c1.raio)  
c2 = Circulo(10)  
print("Raio de c2: ", c2.raio)
```

```
class Circulo:  
  
    raio = None  
  
    def __init__(self, raio=0):  
        self.raio = raio  
  
c1 = Circulo()  
print("Raio de c1: ", c1.raio)  
c2 = Circulo(10)  
print("Raio de c2: ", c2.raio)
```

# Herança

---

```
|class Mae:  
|    nome = 'Joana'  
|    sobrenome = 'Oliveira'  
|    residencia = 'Cachoeirinha'  
  
|class Filha (Mae):  
|    nome = 'Karina'  
|    olhos = 'castanhos'  
  
|class Neta (Filha):  
|    nome = 'Marina'  
  
mae = Mae()  
filha = Filha()  
neta = Neta()  
  
print(mae.nome, filha.nome, neta.nome)  
print(mae.sobrenome, filha.sobrenome, neta.sobrenome)  
print(mae.olhos, filha.olhos, neta.olhos)  
print(mae.residencia, filha.residencia, neta.residencia)
```

# Herança

```
>>> class Mae:  
...     nome = 'Joana'  
...     sobrenome = 'Oliveira'  
...     residencia = 'Cachoeirinha'  
...  
>>> class Filha (Mae):  
...     nome = 'Karina'  
...     olhos = 'castanhos'  
...  
>>> class Neta (Filha):  
...     nome = 'Marina'  
...  
>>> issubclass(Neta, Mae)  
True  
>>> issubclass(Filha, Mae)  
True  
>>> Neta.__bases__  
(<class '__main__.Filha'>,)
```

# Herança

```
>>> class Atlantico:  
...     caracteristica1 = 'É um oceano '  
...  
>>> class Indico:  
...     caracteristica2 = 'perigoso '  
...  
>>> class Pacifico:  
...     caracteristica3 = 'cheio de tsunamis '  
...  
>>> class Artico(Atlantico, Indico, Pacifico):  
...     caracteristica4 = 'e muito gelado!'  
...  
>>> print(Artico.caracteristica1, Artico.caracteristica2, Artico.caracteristica3, Artico.caracteristica4)  
É um oceano perigoso cheio de tsunamis e muito gelado!  
>>> Indico.__bases__  
(<class 'object'>,)  
>>> Pacifico.__bases__  
(<class 'object'>,)  
>>> Artico.__bases__  
(<class '__main__.Atlantico'>, <class '__main__.Indico'>, <class '__main__.Pacifico'>)
```