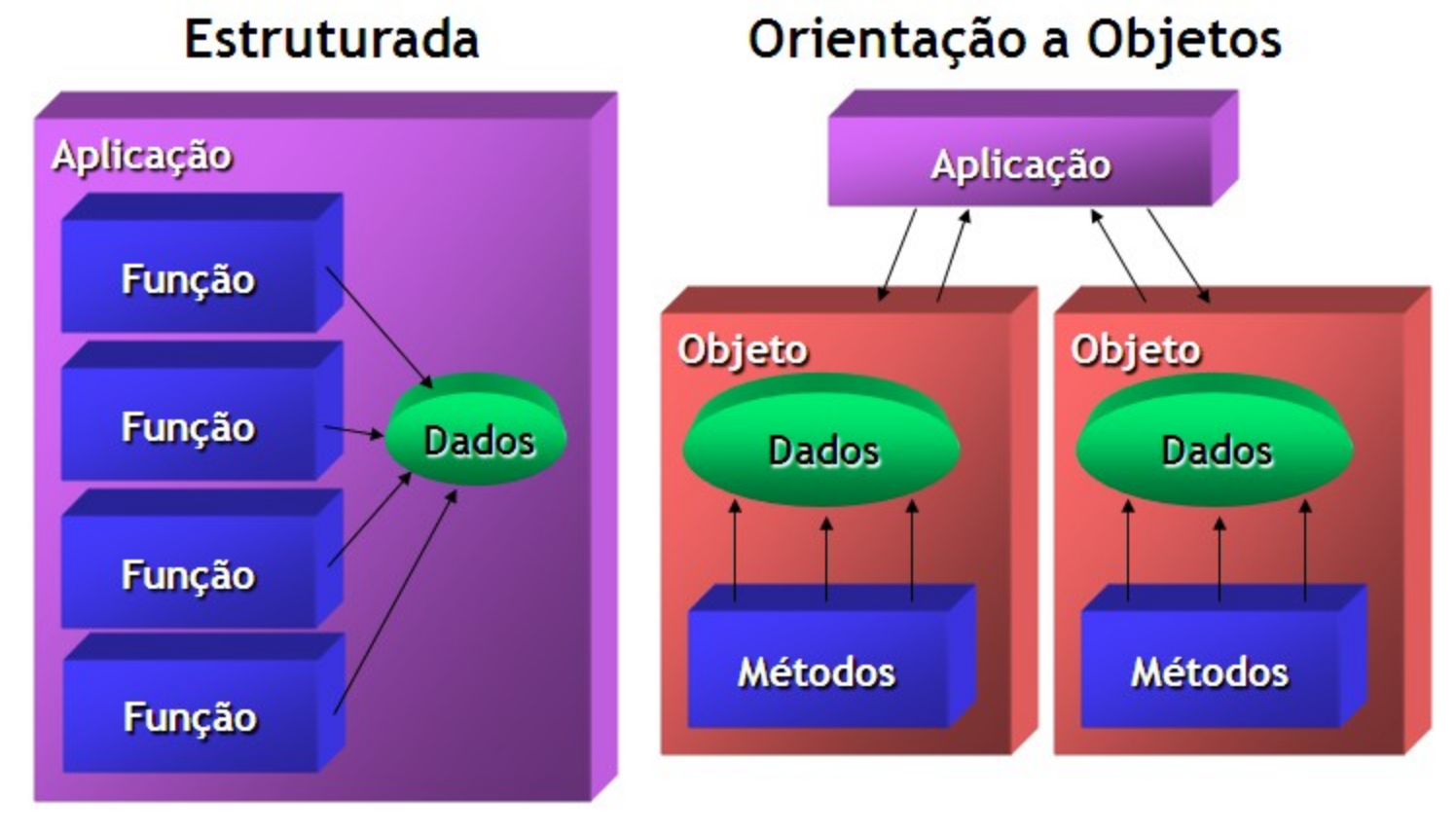
Orientação a objetos



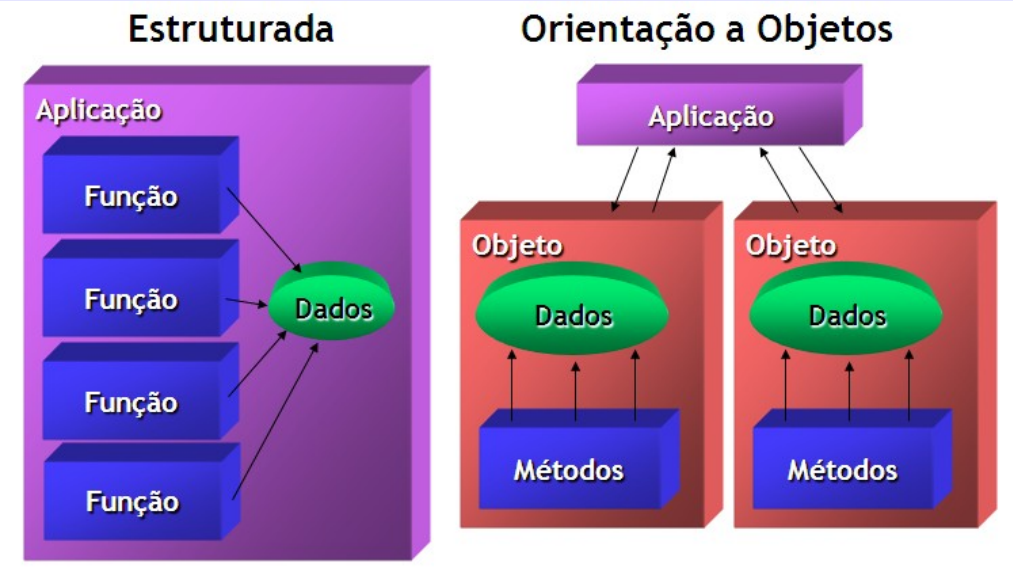
Python OpenCv

Transformar Python em um executavel, instalar pyinstaler:
pip install pyinstaller
Entre no diretorio e execute o comando:
Cria com arquivos de dependêcias o comando abaixo
pyinstaller arquivo.py
Cria somente o .exe
pyinstaller --onefile arquivo.py
Cria somente o .exe e permite escrita
pyinstaller --onefile -w arquivo.py

Orientação a objetos.

O que é um objeto?

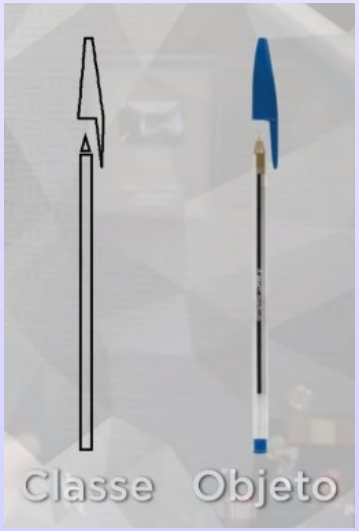
É uma coisa real ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamento e estado atual.



Orientação a objetos.

O que é uma classe?

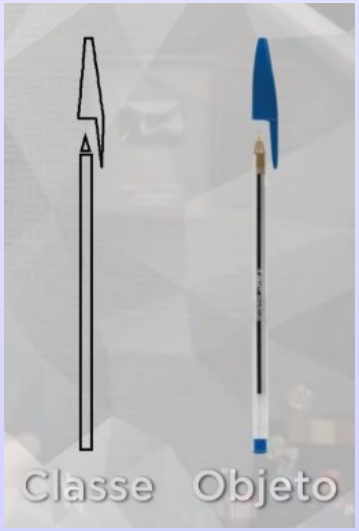
É uma espécie de molde para classificar o objeto.



Orientação a objetos.

Uma classe eu tenho que responder 3 perguntas:

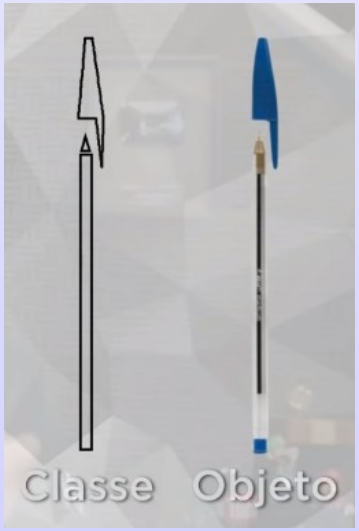
- Coisas que tenho.
- Coisas que eu faço.
- Como estou agora.



Orientação a objetos.

Exemplo de uma classe de um objeto caneta:

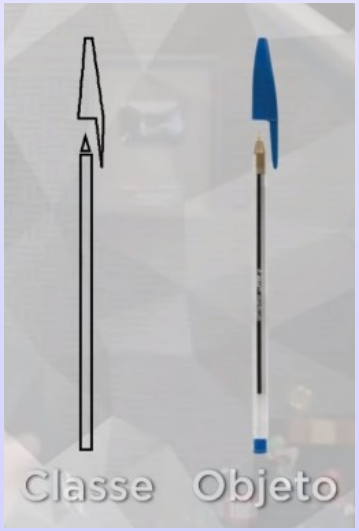
- Que coisas tenho no objeto caneta?
  Modelo , cor, ponta, carga, e tampada
- Que coisas que eu faço?
  Escrever, rabiscar, pintar e tampar
- Como estou agora?



Orientação a objetos.

Exemplo de uma classe de um objeto caneta em POO:

- Atributo, para ficar mais fácil o entendimento pode ser comparado com uma variável na programação estruturada
  Modelo , cor, ponta, carga, e tampada
- Método, para ficar mais fácil o entendimento pode ser comparado com as funções em programação estruturada
  Escrever, rabiscar, pintar e tampar
- Estado.



Orientação a objetos.

Orientação a objetos.

Orientação a objetos.

O que é instanciar?

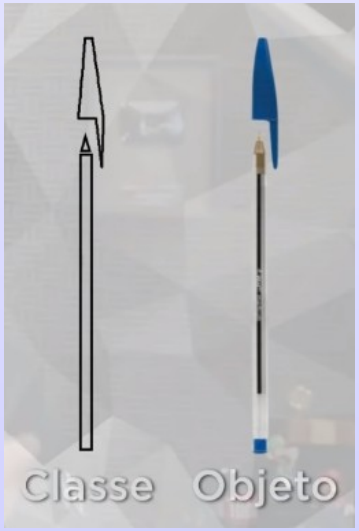
Instanciar é pegar uma classe e transformar em um objeto

Orientação a objetos.

Orientação a objetos.

O que é uma classe?

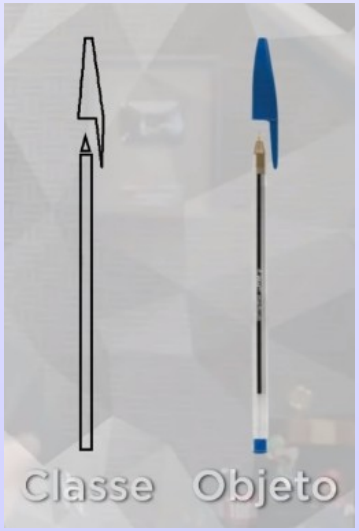
Uma classe define os atributos e métodos comuns que serão compartilhados por um objeto.



Orientação a objetos.

O que é um objeto?

É a instancia de uma classe.



pessoa01 ---- Classe Pessoa.py

```python
class Pessoa:
    #metodo construtor
    def __init__(self, nome, idade, comendo=False, falando=False):
        self.nome = nome
        self.idade = idade
        self.comendo = comendo
        self.falando = falando
    #metodo simples
    def comer(self, alimento):
        if self.comendo:
            print(f'{self.nome}já está comendo.')
            return
        print(f'{self.nome} com a idade de {self.idade} anos, está comendo {alimento}.')
        self.comendo = True
    #metodo simples
    def parar_comer(self):
        if not self.comendo:
            print(f'{self.nome} não esta comendo.')
            return
    #motodo simples
    def falar(self):
        print('A pessoa está falando....')

```

pessoa01 ---- Classe main.py

```
1  from pessoa import Pessoa
2  p1 = Pessoa('John ', 40)
3  p1.parar_comer()
4  p1.comer('Burger')
5  p1.falar()
6
7
```

pessoa02 ---- Classe Pessoa.py

```python
class Pessoa:
    ano_atual = 2024

    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade
    #metodo simples
    def get_ano_nascimento(self):
        print(self.ano_atual - self.idade)
    #O @classmethod permite que você chame o metodo de
    #uma classe sem precisar instânciar.
    #Vale ressaltar que por convenção usamos o cls
    #Para referênciar a classe no classmethod, no
    #lugar onde ficaria o self.
    @classmethod
    def por_ano_nascimento(cls, nome, ano_nascimento):
        idade = cls.ano_atual - ano_nascimento
        return cls(nome, idade)

```

pessoa02 ---- Classe main.py

```
1   from pessoa import Pessoa
2   #Exemplo estanciada a classe
3   p1 = Pessoa('Carlos ', 36)
4   #print(p1)
5   print(p1.nome, p1.idade)
6   p1.get_ano_nascimento()
7   #Não necessita estanciar por causa do @classmethod
8   p2 = Pessoa.por_ano_nascimento('Carlos', 1983)
9   #print(p2)
10  print(p2.nome, p2.idade)
11  p2.get_ano_nascimento()
12
```

pessoa03 ---- Classe Pessoa.py

```python
from random import randint
class Pessoa:
    ano_atual = 2024

    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade
    #metodo simples
    def get_ano_nascimento(self):
        print(self.ano_atual - self.idade)

    @classmethod
    def por_ano_nascimento(cls, nome, ano_nascimento):
        idade = cls.ano_atual - ano_nascimento
        return cls(nome, idade)
    #Um método estático é um método que está vinculado
    #à classe e não ao objeto da classe, ou seja, um método
    # que atua sobre a classe, não sobre as instâncias dela.
    @staticmethod
    def gera_id():
        rand = randint(10000, 19999)
        return rand
```

pessoa03 ---- Classe main.py

```python
from pessoa import Pessoa
p1 = Pessoa('Eder ', 36)
#print(p1)
print(p1.nome, p1.idade)
p1.get_ano_nascimento()
p2 = Pessoa.por_ano_nascimento('Carlos', 1983)
#print(p2)
print(p2.nome, p2.idade)
p2.get_ano_nascimento()
#Conceito de estancia do metodo estatico
print("Gera id direto da classe ",Pessoa.gera_id())
print("Gera id para p1 ",p1.gera_id())
```