



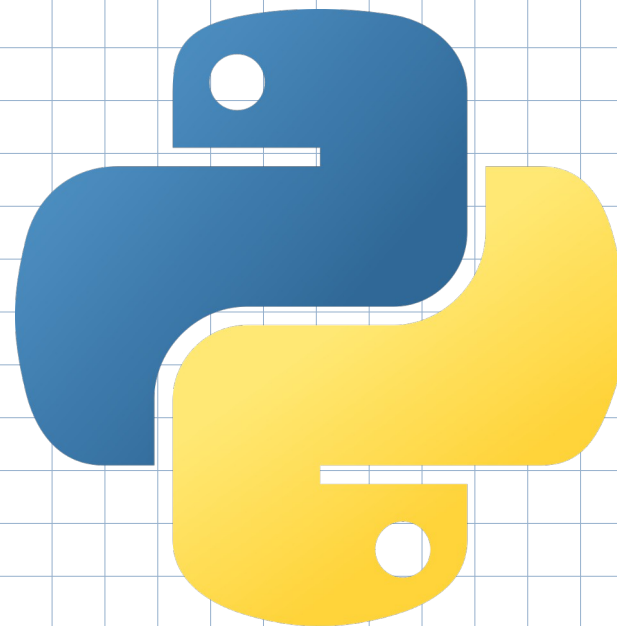
# Programação Python

---

Programação Orientada à Objetos

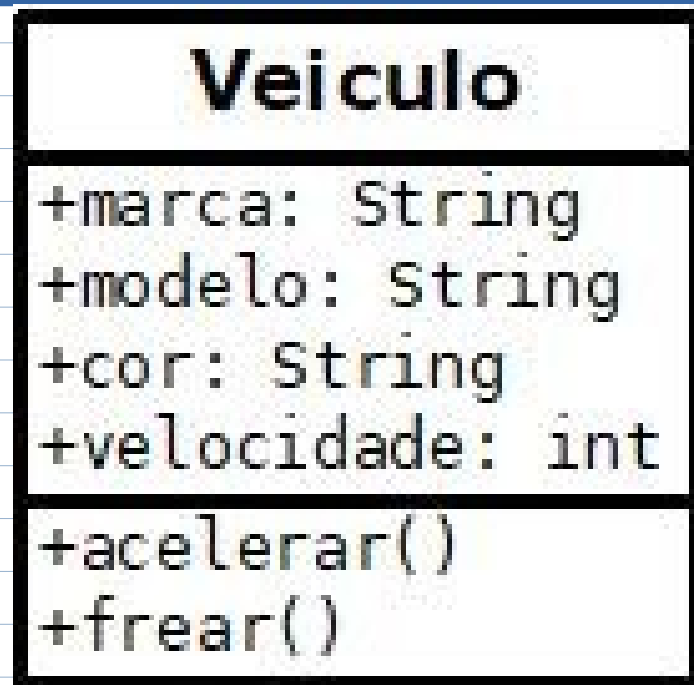
# Tópicos:

- ▼ Introdução à computação
- ▼ Lógica de Programação
- ▼ Introdução ao Python
- ▼ Conceitos de Programação em Python
- ▼ Estruturas de Repetição
- ▼ Estrutura de dados no Python
  - **Programação Orientada à Objetos**
- ▼ Módulos e Pacotes



# Exemplo 1 – Classe Veículo

- Objeto
- Classe
  - Atributo
  - Método
- Encapsulamento



# Programação Orientada à Objetos

---

Implementação da classe Veículo

# Classe Veículo

Veiculo.py ×

```
1 # Classe de teste para veículo gilbertexbom, Today • Initial commit
2 usages  👤 gilbertexbom
3 class Veiculo(object):
4     👤 gilbertexbom
5     def __init__(self, marca, modelo, cor, velocidade):
6         self.marca = marca
7         self.modelo = modelo
8         self.cor = cor
9         self.velocidade = velocidade
```

# Get e Set


```

8
    ① gilbertexbom
9      def setMarca(self, marca):
10         self.marca = marca
11
12      1 usage  ① gilbertexbom
13      def getMarca(self):
14         return self.marca

```

# Get e Set

14

 gilbertexbom


15

```
def setModelo(self, modelo):
```

16

```
    self.modelo = modelo
```

17

```
1 usage  gilbertexbom
```

18

```
def getModelo(self):
```


19

```
    return self.modelo
```

20

# Get e Set

20

 gilbertexbom


21

```
def setCor(self, cor):
```

22

```
    self.cor = cor
```

23

```
1 usage  gilbertexbom
```

24

```
def getCor(self):
```

25

```
    return self.cor
```

26



# Get e Set

```

    gilbertextbom
27  def setVelocidade(self, velocidade):
28      self.velocidade = velocidade gilbertextbom
29
    1 usage  gilbertextbom
30  def getVelocidade(self):
31      return self.velocidade
32
```

# Exibindo os dados do objeto

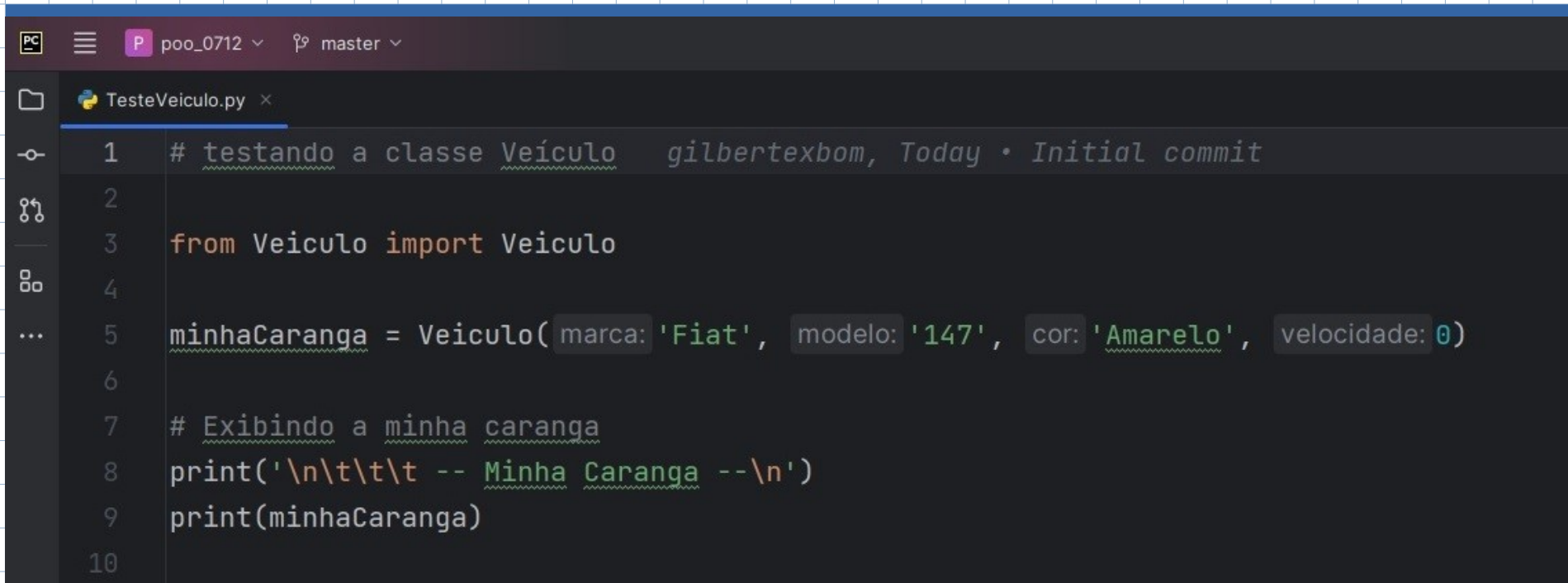
```
32
    gilbertexbom
33     def __str__(self):
34         return(
35             '\n Marca: ' + str(self.getMarca()) +
36             '\n Modelo: ' + str(self.getModelo()) +
37             '\n Cor: ' + str(self.getCor()) +
38             '\n Velocidade: ' + str(self.getVelocidade())
39         )
40
```

# Programação Orientada à Objetos

---

Implementação do arquivo de Teste

# Classe de Teste



```
1 # testando a classe Veiculo gilbertexbom, Today • Initial commit
2
3 from Veiculo import Veiculo
4
5 minhaCaranga = Veiculo(marca: 'Fiat', modelo: '147', cor: 'Amarelo', velocidade: 0)
6
7 # Exibindo a minha caranga
8 print('\n\t\t\t -- Minha Caranga --\n')
9 print(minhaCaranga)
10
```

# Métodos

```
40  
1 usage  👤 gilbertexbom  
41 def acelerar(self):  
42     if self.velocidade < 120:  
43         self.velocidade+=1  
44  
45
```

```
45  
1 usage  👤 gilbertexbom  
46 def frear(self):  
47     if self.velocidade > 0:  
48         self.velocidade-=1
```

# Acelerar

```
10
11 # Acelerando a minha caranga
12 for i in range(0, 200):
13     minhaCaranga.acelerar()
14
15 # Exibindo a minha caranga acelerada
16 print('\n\t\t\t -- Minha Caranga Acelerada --\n')
17 print(minhaCaranga)
```

# Frear

```

18
19     for i in range(0, 200):
20         minhaCaranga.frear()
21
22     # Exibindo a minha caranga freada
23     print('\n\t\t\t -- Minha Caranga Freada --\n')
24     print(minhaCaranga)
25

```

# Exemplo 2 – Classe Elevador

- Objeto
- Classe
  - Atributo
  - Método
- Encapsulamento

Elevador
+andarAtual: int +portaAberta: boolean +peso: float
+subir(andarDesejado) +descer(andarDesejado)



# Programação Orientada à Objetos

---

Implementação da classe Elevador

# Classe Elevador

```
1  # Classe elevador
2
3  2 usages  👤 gilbertexbom +1
4  class Elevador(object):
5
6      # Construtor
7      👤 gilbertexbom
8      def __init__(self, andarAtual, portaAberta, peso):
9          self.andarAtual = andarAtual
10         self.portaAberta = portaAberta
11         self.peso = peso
```

# Get e Set

```
10
11     # Encapsulamento
12     🧑 gilbertexbom
13     def setAndarAtual(self, andarAtual):
14         self.andarAtual = andarAtual
15
16     1 usage 🧑 gilbertexbom
17     def getAndarAtual(self):
18         return self.andarAtual
```

# Get e Set

```
2 usages  👤 gilbertexbom  
18 def setPortaAberta(self, portaAberta):  
19     self.portaAberta = portaAberta  
20  
2 usages  👤 gilbertexbom  
21 def getPortaAberta(self):  
22     return self.portaAberta  
23
```

# Get e Set

23

gilbertexbom

24

```
def setPeso(self, peso):
```

25

```
    self.peso = peso
```

26

1 usage gilbertexbom

27

```
def getPeso(self):
```

28

```
    return self.peso
```

29

# Método para exibir os dados da classe



```
gilbertextbom +1
30 def __str__(self):
31     return(
32         '\n Andar Atual.....' + str(self.getAndarAtual()) +
33         '\n Peso.....{: .2f}'.format(self.getPeso()) +
34         '\n\t\t -- Porta Aberta -- ' if self.getPortaAberta() else '\n\t\t\t -- Porta Fechada -- '
35     )
```

# Programação Orientada à Objetos

---

Implementação do arquivo de Teste

# Criação e instância do objeto da classe elevador

```
1  # Teste Elevador
2  > import ...
5
6  # Criação e instância de um objeto da classe elevador
7  e1 = Elevador( andarAtual: 0, portaAberta: True, random.random() * 1000)
8
9  # Exibindo o elevador parado
10 print('\n\t\t\t -- Elevador Parado --')
11 print(e1)
```



# Método subir

```
36 1 usage  🧑 Gilberto +1
37 def subir(self, andarDesejado):
38     if self.fecharPorta():
39         while self.andarAtual < andarDesejado:
40             self.andarAtual+=1
41             print('{}° andar...'.format(self.andarAtual))
42             self.setPortaAberta(True)
43     else:
44         print('Excesso de Peso! Porta Aberta!')
45
```

# Subindo...

```
12
13  # Subindo...
14  print('\n\t\t\t -- Subir -- ')
15  e1.subir(5)
16
17  # Exibindo o elevador no 5º andar
18  print('\n\t\t\t -- Elevador no 5º andar -- ')
19  print(e1)
```

# Método descender

Tá faltando uma linha de comando!

```
1 usage  🧑 Gilberto
47  def descender(self, andarDesejado):
48      if self.fecharPorta():
49          while self.andarAtual > andarDesejado:
50              self.andarAtual -= 1
51              print('{}ª andar...'.format(self.andarAtual))
52          else:
53      📢 print('Excesso de peso! Porta Aberta!')  Gilberto,
54
```

# Descendo...

```

21  # Descendo...
22  print('\n\t\t\t -- Descer -- ')
23  e1.descer(0)
24  Gilberto, 24 minutes ago • - Commit com os n
25  # Exibindo o elevador no Térreo
26  print('\n\t\t\t -- Elevador no Térreo -- ')
27  print(e1)
28

```

# Controle de Peso

55

2 usages  Gilberto

56

def fecharPorta(self):

57

if self.peso &lt; 750.0:

58

self.setPortaAberta(False)

59

return not self.getPortaAberta()

60

# Exercício 1

- Implemente a classe data de forma a evitar que o usuário da classe crie um objeto inconsistente, ou seja uma data como **32/03** ou **30/02** ou **-01/01...** etc.

**Data**

dia: int

mes: int

ano: int

# Exercício 2

- Crie a classe funcionário e implemente um método para calcular o valor do desconto de INSS com base na seguinte regra:
  - Salário maior do que 5.000, 10%

**Funcionario**

nome: String

salario: float

+calcularINSS()

# Exercício 3

- Implemente a classe Estudante e nela o método que calcula a média das notas bimestrais
  - O ano letivo terá 4 bimestres.

## Estudante

nome: String

notas: float = [4]

+calcularMedia(): float