PROF. INSTRUTOR BELONE DORDETE FRAGA

CLASSES, OBJETOS e MÉTODOS

SATC
EDUCAÇÃO E TECNOLOGIA

Classe

Uma classe é uma forma de definir um tipo de dado em uma **linguagem orientada a objeto**. Ela é formada por dados e comportamentos.

Classe

Para definir os dados utilizamos os atributos, e para definir o comportamento são utilizados métodos. Depois que uma classe é definida podem ser criados diferentes objetos que utilizam a classe.

Classe

Representação do cachorro Bilú:

- **Propriedades** : [Cor do corpo : castanha; cor dos olhos : pretos; altura: 15 cm;   comprimento: 38 cm largura : 24 cm]
- **Métodos** : [balançar o rabo , latir , correr, deitar , sentar]

SATC
EDUCAÇÃO E TECNOLOGIA

DECLARAR CLASSE

```
class Pessoa:
    pass
```

PS: Deste modo no arquivo pessoa.py

Objetos

Um objeto é qualquer coisa, real ou abstrata, sobre a qual armazenamos dados e realizamos operações que manipulam tais dados.

Objetos

Um objeto de uma classe possui:

- Atributos à características ou propriedades que definem o objeto.

- Comportamento à conjunto de ações pré-definidas (métodos)

Objetos

- Pássaro

Características:

cores
forma do bico
tipo do vôo

Comportamento:

voar()
piar()

- Pessoa

Características:

cor dos olhos: azuis
data nascimento: 16/02/70
peso: 70kg
altura: 1,70m

Comportamento:

andar
falar
comer
rir

ATRIBUTOS

Os atributos podem ser definidos direto dentro da classe (Atributo de Classe) ou dentro dos métodos criados na classe (Atributos da Instância).

A diferença é que o atributo da classe pode ser acessado e alterado pela classe mas as instâncias só podem acessá-lo mas não conseguem alterá-lo.

__dict__ Não é um método - é um atributo - e é a forma implementada, oficialmente, na linguagem de guardar atributos de instância nos objetos.

```python
class Cls:
    num = 1

inst = Cls()
print(inst.__dict__)
# {} (o objeto ainda não possui nenhum atributo de instância...)
print(inst.num)
# 1 (e como não possui o atributo "num" recorre um nivel acima onde encontra o atributo da classe...)
inst.num = 2
print(inst.__dict__)
# {'num': 2} (depois da atribuição agora o objeto possui seu próprio atributo...)
print(inst.num)
# 2 (então não acessa mais o atributo de mesmo nome da classe...)
print(Cls.num)
# 1 (onde o atributo ainda possui valor inicial.)
```

PROF. INSTRUTOR BELONE DORDETE FRAGA

Métodos

MÉTODOS DE INSTÂNCIA

Qualquer método criado dentro da classe, que ao ser instanciada o objeto vai receber os métodos dela.

Todos métodos de instância da classe recebem como primeiro atributo o **self**, ele refere-se a instância que foi criada a partir da classe.

SATC
EDUCAÇÃO E TECNOLOGIA

```python
class Pessoa:
    def comer(self, alimento):
        print(f'Pessoa está comendo {alimento}.')
```

MÉTODOS DE CLASSE

São métodos próprios da classe onde ficam acessíveis apenas para a classe em si e não para objetos instanciados a partir da classe.

Para esse comportamento devemos passar antes um decorator chamado **@classmethod** e o primeiro parâmetro passa a ser a própria classe **(cls)**.

SATC
EDUCAÇÃO E TECNOLOGIA

```python
class Pessoa():
    def __init__(self, altura, idade):
        self.altura = altura
        self.idade = idade



class Aluno():
    def __init__(self, altura, idade):
        self.altura = altura
        self.idade = idade

    @classmethod
    def construir_aluno_pessoa(cls, pessoa):
        return cls(pessoa.altura, pessoa.idade)

    def estudar(self):
        print("Estou estudando")


joao = Pessoa(1.85, 18)
mariaAluna = Aluno(1.68, 18)
mariaAluna.estudar()

# Construindo classe Aluno recebendo uma instancia da classe pessoa
joaoAluno = Aluno.construir_aluno_pessoa(joao)
joaoAluno.estudar()
```

```
class Pessoa():
  def __init__(self, altura, idade):
    self.altura = altura
    self.idade = idade

class Aluno():
  def __init__(self, altura, idade):
    self.altura = altura
    self.idade = idade

  @classmethod
  def construir_aluno_pessoa(cls, pessoa):
    return cls(pessoa.altura, pessoa.idade)

  def estudar(self):
    print('Estudando')

joao = Pessoa(1.80, 20)
mariaAluno = Aluno(1.60, 18)
mariaAluno.estudar()

joaoAluno = Aluno.construir_aluno_pessoa(joao)
joaoAluno.estudar()
print(joaoAluno.altura, joaoAluno.idade)
```

MÉTODOS ESTÁTICOS

São métodos que não recebem o contexto da instância e da classe, poderiam até ser criados fora da classe.
Deve-se adicionar um decorator @staticmethod antes da assinatura do método.

```python
from random import randint
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    @staticmethod
    def gera_id():
        rand = randint(10000, 19999)
        return rand
```

MÉTODOS ABSTRATOS

São métodos que não possuem um corpo apenas a assinatura do método, geralmente são criados dentro de uma classe abstrata para serem sobrescritos nas instâncias que herdarem dessa classe.

```python
from abc import ABC, abstractmethod
class Example(ABC):
    @abstractmethod
    def sacar(self, valor):
        pass
```

MÉTODOS "MÁGICOS"

São métodos especiais que você pode definir para adicionar "magia" às classes. Eles estão sempre cercados por dois underlines (por exemplo, __init__ ou __lt__).

MÉTODOS "MÁGICOS"

__NEW__

É o primeiro método a ser chamado na instanciação de um objeto. Ele recebe a classe e, a seguir, quaisquer outros argumentos e passará para o __init__.

Use o **__new__** quando você precisar controlar a criação de uma nova instancia da classe.
Use o **__init__** quando você precisar controlar a inicialização de uma nova instancia.

SATC
EDUCAÇÃO E TECNOLOGIA

```python
class Pessoa:
    def __new__(cls, *args, **kwargs):
        pass
```

MÉTODOS "MÁGICOS"

__INIT__

O inicializador da classe, é chamado assim que a classe for instanciada.

```python
class Pessoa:
    def __init__(self, nome):
        self.nome = nome
```

MÉTODOS "MÁGICOS"

__CALL__

Permite que uma instância de uma classe seja chamada como uma função. Essencialmente, isso significa que **Pessoa()** é o mesmo que **Pessoa.__call__()**.

SATC
EDUCAÇÃO E TECNOLOGIA

```python
class Pessoa:
    def __call__(self, *args, **kwargs):
        print(args)
        print(kwargs)

p = Pessoa()
p(1, 2, 3, nome='Gui')
# (1, 2, 3)
# {'nome': 'Gui'}
```

__call__ leva um número variável de argumentos, isso significa que você define **__call__** como faria com qualquer outra função, usando quantos argumentos quiser.

EXERCICIO: 01

1. **Classe Triangulo:** Crie uma classe que modele um triangulo:
   - Atributos: LadoA, LadoB, LadoC
   - Métodos: calcular Perímetro, getMaiorLado;

Crie um programa que utilize esta classe. Ele deve pedir ao usuário que informe as medidas de um triangulo. Depois, deve criar um objeto com as medidas e imprimir sua área e maior lado.

EXERCICIO: 02

2. **Classe Funcionário:** Implemente a classe Funcionário. Um funcionário tem um nome e um salário. Escreva um construtor com dois parâmetros (nome e salário) e o método aumentarSalario (porcentualDeAumento) que aumente o salário do funcionário em uma certa porcentagem. Exemplo de uso:

```
harry = funcionario("harry", 2500)
harry.aumentaSalario(10)
```

Faça um programa que teste o método da classe.

EXERCICIO: 03

3. Crie uma classe Livro que possui os atributos nome, qtdPaginas, autor e preço.
   - Crie os métodos getPreco para obter o valor do preco e o método setPreco para setar um novo valor do preco.

Crie um codigo de teste

EXERCICIO: 04

4. Implemente uma classe Aluno, que deve ter os seguintes atributos: nome, curso, tempoSemDormir (em horas). Essa classe deverá ter os seguintes métodos:

- estudar (que recebe como parâmetro a qtd de horas de estudo e acrescenta tempoSemDormir)
- Dormir (que recebe como parâmetro a qtd de horas de sono e reduz tempoSemDormir)

Crie um código de teste da classe, criando um objeto da classe aluno e usando os métodos estudar e dormir. Ao final imprima quanto tempo o aluno está sem dormir

EXERCICIO: 05

5. **Classe carro:** Implemente uma classe chamada Carro com as seguintes propriedades:

- Um veículo tem um certo consumo de combustível (medidos em km / litro) e uma certa quantidade de combustível no tanque.
- O consumo é especificado no construtor e o nível de combustível inicial é 0.
- Forneça um método andar( ) que simule o ato de dirigir o veículo por uma certa distância, reduzindo o nível de combustível no tanque de gasolina. Esse método recebe como parâmetro a distância em km.
- Forneça um método obterGasolina( ), que retorna o nível atual de combustível.
- Forneça um método adicionarGasolina( ), para abastecer o tanque.
- Faça um programa para testar a classe Carro. Exemplo de uso:

```
meuFusca = Carro(15); # 15 quilômetros por litro de combustível.
meuFusca.adicionarGasolina(20); # abastece com 20 litros de combustível.
meuFusca.andar(100); # anda 100 quilômetros.
meuFusca.obterGasolina() # Imprime o combustível que resta no tanque.
```

EXERCICIO: 06

6. Crie uma classe Aluno, que possui como atributo um nome e cpf. Crie outra classe chamada Equipe, que possui como atributo uma lista de participantes do tipo Aluno e outro atributo chamado projeto.

Crie uma terceira classe chamada GerenciadorEquipes. Essa classe possui como atributo uma lista de todas as equipes formadas. Ela deverá possuir o método criarEquipe, que recebe uma lista de alunos de uma equipe e diz se a equipe pode ser formada ou não. Caso não haja nenhum aluno da equipe a ser formada em uma outra equipe com o mesmo projeto, então a equipe é criada e acrescentada à lista. Caso contrário é informada que a equipe não pode ser criada.