Insper

Design de Software

Aula: Orientação a Objetos em Python

2020 – Engenharia

Paradigmas de programação

- A forma básica de programação em Python é a **programação procedural**:
  - O elemento central da estruturação de código é a **função**
  - A computação acontece por meio de **chamadas de função**
  - E por sequências de operações elementares em Python.

Paradigmas de programação

- Com as funcionalidades de ‘if’, ‘while’, e ‘for’, e na ausência do comando ‘goto’, passamos a praticar **programação estruturada**
    - O fluxo de execução é controlado por ‘if’, ‘while’, e ‘for’;
    - ... mais tudo da programação procedural

Paradigmas de programação

- Próxima etapa seria a **programação orientada a objetos**:
  - O elemento central da estruturação de código é a **classe**
  - A computação acontece por meio de criação de **objetos** e da interação entre eles
  - ... mais tudo da programação procedural

Programação Orientada a Objetos

Paradigma de programação onde dados e funcionalidades são organizados de forma conjunta em classes e objetos.

Dados + Código = Objetos

Classe = tipo do Objeto

Objeto = instância da Classe

Exemplo

Scratch é uma linguagem orientada a objetos

Objetos são usados a todo momento em Python

Um **objeto** chamado **A**,
do tipo **list**

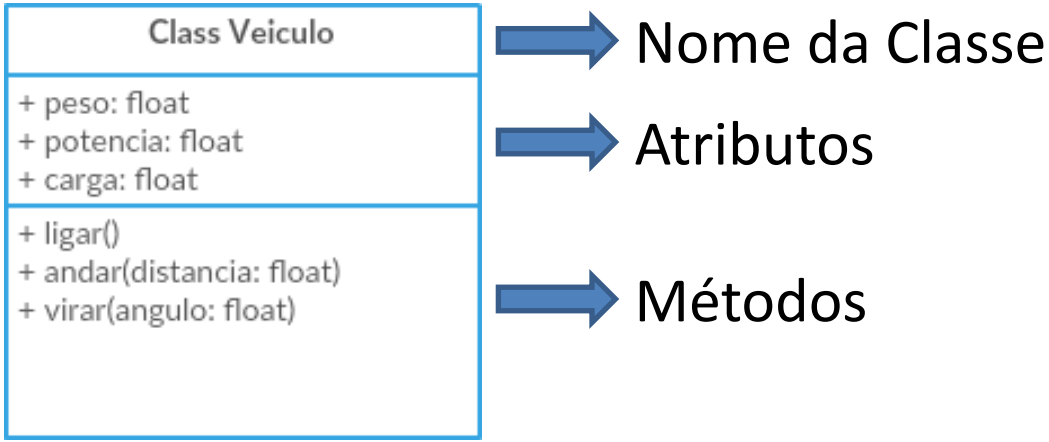
Nome da **classe**

```
A = list()  # Equivale a A = []

A.append(42)
A.append(57)
print(A)

A.reverse()
print(A)

A.sort()
print(A)

print(A.index(57))
```

Chamadas de **método**

Vamos pensar em classes

- Ferramenta: Diagrama de Classes



Fonte: Wikipedia

Vamos criar um objeto

Até agora usamos objetos já criados para nós no Python, vamos criar nossas próprias classes e objetos.

Para criar um classe em Python usamos a instrução:

```
class
```

Um exemplo: primeira classe

```
1  class Point:
2      """ Classe que representa um Ponto"""
3      pass
```

- Não é uma função!
- Point é uma classe
- Classes começam com maiúsculas, de acordo com a PEP 08

Usando o primeiro objeto

```
1  class Point:
2      """ Classe que representa um ponto """
3      pass
4
5  # Criando objetos
6  p1 = Point()
7  p2 = Point()
8
9  # Verificando tipo
10 print("Tipo de p1: {0}".format(type(p1)))
11
12 # Imprimindo o ponto
13 print(p1)
```

```
Tipo de p1: <class '__main__.Point'>
<__main__.Point object at 0x10cc21990>
```

Melhoria: função para criar atributos

```
 1  class Point:
 2      """ Classe que representa um Ponto"""
 3      pass
 4  # Cria os atributos x e y
 5  def init(point, vx, vy):
 6      point.x = vx
 7      point.y = vy
 8
 9  # Criando objetos
10  p1 = Point()
11  p2 = Point()
12
13  init(p1, 2, 3)
14  init(p2, 4, 5)
```

E os dados do Ponto?

Não seria ideal poder criar o ponto assim? (já com os valores iniciais)

```
p1 = Point(2,3)
p2 = Point(4,5)
```

Construtores

são dois underscore

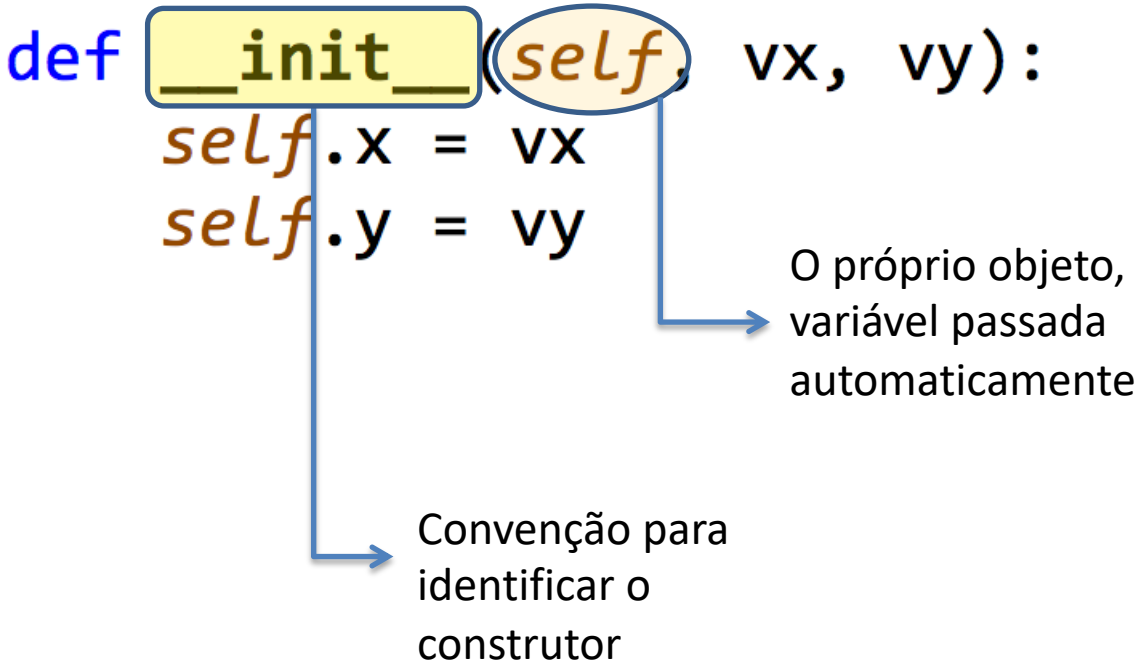
```
1  class Point:
2      """ Classe que representa um Ponto"""
3
4      # Cria os atributos x e y
5      def __init__(self, vx, vy):
6          self.x = vx
7          self.y = vy
8
9  # Criando objetos
10 p1 = Point(2, 3)
11 p2 = Point(4, 5)
```

E os dados do objeto?

```python
# Imprimindo o atributo x do objeto p1
print(p1.x)

# Alterando o atributo x do objeto p1
p1.x = 4

# E agora?
print(p1.x)
```

```
2
4
```

Método construtor



Método?

Em orientação a objetos:

método = função que pertence ao objeto.

Por exemplo:

```
lista.append(3)
linha.strip()
```

Um método comum

Distância do ponto a outro ponto.

```
def distance_to(self, other_point):
    """ calcula a distância entre o ponto
     e o outro ponto (other_point) """
    return sqrt((self.x - other_point.x)**2 + \
        (self.y - other_point.y)**2)
```

Tudo junto

```python
from math import sqrt

class Point:
    """ Classe que representa um Ponto"""

    # Cria os atributos x e y
    def __init__(self, vx, vy):
        self.x = vx
        self.y = vy

    def distance_to(self, other_point):
        """ calcula a distância entre o ponto
         e o outro ponto (other_point) """
        return sqrt((self.x - other_point.x)**2 + \
            (self.y - other_point.y)**2)
```

Usando a classe criada

```
# Criando objetos
p1 = Point(2,3)
p2 = Point(4,5)

# Usando o método distance_to
dist = p1.distance_to(p2)
```

Exercícios

1. Crie uma classe chamada Retangulo, com um __init__ que pede dois pontos:

- Ponto do canto inferior esquerdo
- Ponto do canto superior direito

Pede-se:

a) adicione um método que calcule o perímetro.

b) adicione um método que calcula a área.

Escreva um código que demonstre o funcionamento de sua classe

Solução

exemplo de execução:

```
pa = Point(2, 3)
pb = Point(4, 5)

r = Retangulo(pa, pb)

print(r.perimetro())
print(r.area())
```

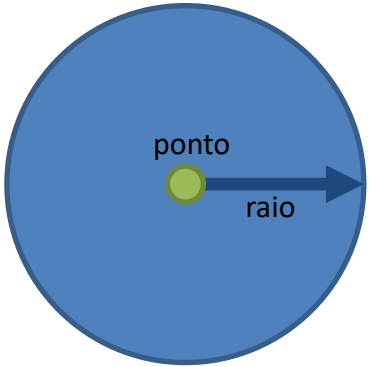
código:

```
from math import sqrt

class Point:
    """ Classe que representa um Ponto. """
    def __init__(self, vx, vy):
        self.x = vx
        self.y = vy

    def distance_to(self, other_point):
        """ Calcula a distância entre o próprio ponto e o outro ponto."""
        return sqrt((self.x - other_point.x)**2 + (self.y - other_point.y)**2)

class Retangulo:
    """ Classe que representa um Retângulo. """
    def __init__(self, p1, p2):
        self.pie = p1
        self.psd = p2

    def perimetro(self):
        a = self.psd.x - self.pie.x
        b = self.psd.y - self.pie.y
        return 2 * (a + b)

    def area(self):
        a = self.psd.x - self.pie.x
        b = self.psd.y - self.pie.y
        return a * b
```

Exercícios

2. Crie uma classe chamada Circulo que tem um ponto como centro e um valor de raio.

a) adicione um método que avalia se um algum objeto Ponto está dentro da área do círculo ou não



Solução

exemplo de execução:

```
c = Point(2, 3)
R = 10

p = Point(4, 5)
q = Point(1000, 2000)

circ = Circulo(c, R)

print(circ.dentro(p))
print(circ.dentro(q))
```

código:

```
from math import sqrt

class Point:
    """ Classe que representa um Ponto. """
    def __init__(self, vx, vy):
        self.x = vx
        self.y = vy

    def distance_to(self, other_point):
        """ Calcula a distância entre o próprio ponto e o outro ponto."""
        return sqrt((self.x - other_point.x)**2 + (self.y - other_point.y)**2)

class Circulo:
    """ Classe que representa um Círculo. """
    def __init__(self, centro, raio):
        self.centro = centro
        self.raio = raio

    def dentro(self, ponto):
        if self.centro.distance_to(ponto) < self.raio:
            return True
        else:
            return False
```

Insper

www.insper.edu.br