Disciplina: IPOO - Introdução à Programação Orientada a Objetos

Prof. Luiz Cláudio Dalmolin - luiz.dalmolin@udesc.br

Orientação a Objetos com Python:

Criação de classes, objetos e envio de mensagens (chamada de métodos)

In [1]:
```
# exxemplo inicial

class TV:
    def __init__(self,vol,c):
        self.volume = vol
        self.canal = c

    def mostraCanal(self):
        return self.canal

    def mudaCanal(self,c):
        self.canal = c
##
tv1 = TV(1,2)
print(tv1.mostraCanal())
tv1.mudaCanal(3)
print(tv1.mostraCanal())
x = TV(0,0)
print(x.mostraCanal())
```

```
2
3
0
```

In [4]:
```
# exxemplo inicial

class TV:
    def __init__ (self,vol,c):
        self.volume = vol
        self.canal = c

    def mostraCanal(self):
        return self.canal

    def mudaCanal(self,c):
        self.canal = c
##
tv1 = TV(1,2)
print(tv1.mostraCanal())
tv1.mudaCanal(3)
print(tv1.mostraCanal())
x = TV(0,0)
print(x.mostraCanal())
```

```
2
3
0
```

In [1]:
```
class TV:
    ligada = None            # Opcional, apenas facilita a identificação dos atr
    canal = None
    def __init__ (self):     # método construtor; self é um objeto TV em si
        self.ligada = False  # é um atributo de self, ou seja, do objeto TV
        self.canal = 2       # ao especificar atributos do objeto, sempre usar s
#######
tv = TV()                    # cria-se um objeto associado a variável tv utiliza
print(tv.ligada)             # ou seja, tv guarda uma instância (objeto) de TV
print(tv.canal)

tv_sala = TV()               # cria-se outro objeto, associado à variável tv_sal
tv_sala.ligada = True
tv_sala.canal = 4

print(tv.ligada)
print(tv.canal)
print(tv_sala.ligada)
print(tv_sala.canal)
```

```
False
2
False
2
True
4
```

Classe e Objetos

1. Adicione ao exemplo anterior os atributos tamanho e marca à classe TV. Crie dois objetos TV e atribua tamanhos e marcas diferentes. Depois, imprima o valor desses atributos de forma a confirmar a independência dos valores de cada instância (objeto).

In [5]:
```
class TV:
    ligada = None            # Opcional, apenas facilita a identificação dos atr
    canal = None
    tamanho = None
    marca = None
    def __init__ (self):     # método construtor; self é um objeto TV em si
        self.ligada = False  # é um atributo de self, ou seja, do objeto TV
        self.canal = 2       # ao especificar atributos do objeto, sempre usar s
        self.tamanho = 40
        self.marca = "Panasonic"
        #######
tv1 = TV()                    # cria-se um objeto associado a variável tv utiliz
print(tv1.ligada)             # ou seja, tv guarda uma instância (objeto) de TV
print(tv1.canal)
print(tv1.tamanho)
print(tv1.marca)

tv2 = TV()                # cria-se outro objeto, associado à variável tv_sala
tv2.ligada = True
tv2.canal = 4
print(tv2.canal)
```

```
print(tv2.marca)

print(tv1.marca)
```

```
False
2
40
Panasonic
4
Panasonic
Panasonic
```

In [6]:
```
# adicionando comportamento à classe
class TV:
    ligada = None
    canal = None
    def __init__(self):
        self.ligada = False
        self.canal = 2
    def muda_canal_para_baixo(self):
        self.canal -= 1
    def muda_canal_para_cima(self):
        self.canal += 1
#####
tv = TV()
tv.muda_canal_para_cima()
tv.muda_canal_para_cima()
print(tv.canal)
tv.muda_canal_para_baixo()
print(tv.canal)
x1 = TV()
x1.muda_canal_para_cima()
x1.muda_canal_para_cima()
x1.muda_canal_para_cima()
x1.muda_canal_para_cima()
x1.muda_canal_para_cima()
print("Canal = ", x1.canal)
```

```
4
3
Canal =  7
```

In [7]:
```
# passagem de parâmetros
class TV:
    ligada = None
    canal = None
    cmin = None
    cmax = None

    def __init__(self, cc, min, max):
        self.ligada = False
        self.canal = cc
        self.cmin = min
        self.cmax = max

    def muda_canal_para_baixo(self):
        if(self.canal - 1 >= self.cmin):
            self.canal -= 1


    def muda_canal_para_cima(self):
```

```
        if (self.canal + 1 <= self.cmax):
            self.canal += 1
        if (self.canal == self.cmax):
            self.canal = self.cmin

########
tv1 = TV(5,2,60)

for x in range(0,120):
      tv1.muda_canal_para_cima()
print(tv1.canal)

for x in range(0,120):
      tv1.muda_canal_para_baixo()

print(tv1.canal)
```

```
9
2
```

2. Atualmente, a classe TV inicializa o canal com 2. Modifique a classe TV de forma a receber o canal inicial em seu construtor.

In [ ]:
```
# alterado no exemplo acima.
```

3. Modifique a classe TV de forma que, se pedir para mudar o canal para baixo, além do mínimo, vá para o canal máximo. Se mudar para cima, além do canal máximo, que volte ao canal mínimo. Exemplo:

   tv = TV(2,10)

   tv.muda_canal_para_baixo()

   tv.canal

10

tv.muda_canal_para_cima()

tv.canal

2

In [ ]:
```
# alterado no exemplo acima.
```

In [15]:
```
# Crie uma classe Pessoa, com os atributos nome e idade e métodos para atribuir
# conteúdo aos atributos e para ler os mesmos.
# - na execução, crie objetos do tipo Pessoa, defina seus atributos e imprima
# os mesmos na tela.

class Pessoa:
    def __init__ (self, nome="", idade=0):
        self.nome = nome
        self.idade = idade
```

```
    def setNome(self, nome):
        self.nome = nome

    def setIdade(self, idade):
        self.idade = idade

    def getNome(self):
        return self.nome

    def getIdade(self):
        return self.idade

###
p1 = Pessoa()
p1.setNome("Carlos")
p2 = Pessoa()
p2.setIdade(30)
p1.setIdade(23)
print (p1.getNome() + " tem " + str(p1.getIdade()) + " anos")
print (p2.getNome() + " tem " + str(p2.getIdade()) + " anos")
print (p1.getIdade())
print(p2.getIdade())
```

```
Carlos tem 23 anos
 tem 30 anos
23
30
```

In [8]:

```
# Altere o exercício anterior para criar uma lista de objetos do tipo Pessoa,
# defina seus atributos e imprima os mesmos na tela.

class Pessoa:
    def __init__ (self, nome="", idade=0):
        self.nome = nome
        self.idade = idade

    def setNome(self, nome):
        self.nome = nome

    def setIdade(self, idade):
        self.idade = idade

    def getNome(self):
        return self.nome

    def getIdade(self):
        return self.idade

###
P = []                              # cria uma lista vazia chamada 'P'
while True:                         # cria um laço de repetição
    nome1=input("Digite nome da pessoa para cadastrar: ('fim' para sair): ")
    if nome1 == "fim":
        break
    idade1=int(input("Digite idade de %s: " % nome1))
    p1 = Pessoa()                   # cria objeto da classe Pessoa (instancia) e
    p1.setNome(nome1)               # chama o método setNome para definir o atri
    p1.setIdade(idade1)             # chama o método setIdade para defnir o atri
    P.append(p1)                    # inclui o objeto Pessoa 'p1' (com seus atri
```

```
print("")

#for x, e in enumerate(P):           # cria um laço que percorre toda a lista 'P
for e in P:
    print(e.getNome() + " tem " + str(e.getIdade()) + " anos") # para cada objet
```

```
Digite nome da pessoa para cadastrar: ('fim' para sair): Alvaro
Digite idade de Alvaro: 18
Digite nome da pessoa para cadastrar: ('fim' para sair): Ana
Digite idade de Ana: 29
Digite nome da pessoa para cadastrar: ('fim' para sair): Pedro
Digite idade de Pedro: 11
Digite nome da pessoa para cadastrar: ('fim' para sair): fim

Alvaro tem 18 anos
Ana tem 29 anos
Pedro tem 11 anos
```

O programa abaixo cria a define a classe Calculadora e os métodos dos objetos dessa classe. Após, cria um objeto da classe Calculadora (armazenado na variável 'c') e envia mensagens ao mesmo, solicitando a execução de métodos de instância (ou de objetos) para o objeto do tipo Calculadora.

In [17]:
```
#calculadora.py
class Calculadora:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def soma(self):
        return self.a + self.b
    def subtrai(self):
        return self.a - self.b
    def multiplica(self):
        return self.a * self.b
    def divide(self):
        return self.a / self.b
## ---

c = Calculadora(12,2)   # criando o objeto da classe Calculadora e atribuindo o
# enviando médotos ao objeto armazenado em 'c' e imprimindo os resultados da exe
print('Soma:', c.soma())
print('Subtração:', c.subtrai())
print('Multiplicação:', c.multiplica())
print('Divisão:', c.divide())
```

```
Soma: 14
Subtração: 10
Multiplicação: 24
Divisão: 6.0
```

In [31]:
```
# forma inconveniente porque todos os objetos caneta terão os
# mesmos "estados" (azul, simples, 5.0)
class Caneta:
    cor = "azul"
    tipo = "simples"
    preco = 5.0

    def mostraPreco(self):
```

```
        return self.preco
###
c1 = Caneta()
print(c1.mostraPreco())
```

```
5.0
```

Exercício

Criar uma classe Aluno com atributos matricula, nome, n1, n2. Criar métodos para definir e retornar atributos e calcular média de notas de um aluno.

In [ ]:

In [13]:
```
class Retangulo:
    def __init__(self,comp,larg):
        self.comprimento = comp
        self.largura = larg
    def mudarValorLados(self,comp,larg):
        self.comprimento = comp
        self.largura = larg
    def informarDados(self):
        return ("Comprimento:",self.comprimento, "Largura:",self.largura)
    def calcularArea(self):
        return self.comprimento * self.largura
    def perimetro(self):
        return self.comprimento * 2 + self.largura * 2
###
r = Retangulo(10,20)
print("Area = ",r.calcularArea())
print(r.perimetro())
print(r.informarDados())
```

```
Area =  200
60
('Comprimento:', 10, 'Largura:', 20)
```

In [7]:
```
# Exercícios com Strings
s = "Olá, mundo!"           # Imprimindo uma string.
print(s)
print(type(s))                    # Tipo de uma string.
print(len(s))                     # Tamanho de uma string.
print("Meu Brasil " + "brasileiro")   # Concatenação
s1 = s.replace("mundo", "meu abacate") # Substitui uma substring por alguma
                                       # outra coisa.
print(s1)
print(s.startswith("Olá"))   # A string s começa com "Olá"?
print(s.endswith("mundo"))  # A string s termina com "mundo"?
print(s1.count("abacate"))  # Quantas ocorrências da palavra "abacate" a string

# procurando uma substring em uma string
string = "Universidade do Estado de Santa Catarina"
substring = "Estado"
print(substring in string) # retorna True se achou
```

```
if substring in string:
    print("Encontrou!")
```

```
Olá, mundo!
<class 'str'>
11
Meu Brasil brasileiro
Olá, meu abacate!
True
False
1
True
Encontrou!
```

In [3]:
```
# Exercícios com Strings
# cadastrando strings e procurando por substring
L = []
while True:
    a = input("Digite um nome: ")
    if a == "fim":
        break
    L.append(a)

y = input("Procurar por: ")
for x, e in enumerate(L):
    if y in e:
        print("na posição %d foi encontrado o nome '%s'" % (x, e))
#---
```

```
Digite um nome: Alvaro Luiz da Silva
Digite um nome: Maria Joaquina
Digite um nome: Carlos da Silva Tavares
Digite um nome: José Silva Antunes
Digite um nome: Maria Francisca
Digite um nome: fim
Procurar por: Silva
na posição 0 foi encontrado o nome 'Alvaro Luiz da Silva'
na posição 2 foi encontrado o nome 'Carlos da Silva Tavares'
na posição 3 foi encontrado o nome 'José Silva Antunes'
```

In [1]:
```
class Aluno:
    matricula = None
    nome = None
    def __init__ (self, mat, nom):
        self.matricula = mat
        self.nome = nom
    def getMatricula(self):
        return self.matricula
    def getNome(self):
        return self.nome
###
L = []
while True:
    n = input("Digite o nome: ")
    if n == "fim":
        break
    m = input("Matrícula: ")
    a1 = Aluno(m,n)
    L.append(a1)

p=input("procurar por: ")
```

```
for x, e in enumerate(L):
#    if p in e.getNome() or p in e.getMatricula():
    if e.getNome().find(p) or e.getMatricula().find(p):
        print("Matrícula: %s - Nome: %s" % (e.getMatricula(), e.getNome()))
```

```
Digite o nome: abcd
Matrícula: 333
Digite o nome: 33 luiz
Matrícula: 421
Digite o nome: teste
Matrícula: 0339
Digite o nome: fim
procurar por: 33
Matrícula: 333 - Nome: abcd
Matrícula: 421 - Nome: 33 luiz
Matrícula: 0339 - Nome: teste
```