Universidade Federal de Uberlândia

Faculdade de Engenharia Elétrica
Programação Script
Prof. Felipe A. Louza

Lista 12

Módulos em Python; Classes

Questão 1

Assinale as alternativas **corretas**?

(a) Módulos podem se tornar uma entrada para o interpretador **Python** e assim ele é chamado de script.

(b) Módulos é o mesmo que funções.

(c) Módulos servem para organizar melhor nosso código.

(d) Módulos são compostos por funções (definições) e podem ter outros comandos (statements).

Questão 2

Dado o arquivo `fatorial.py` com as seguintes linhas de código. Como podemos utilizar as funções `fat1()`, `fat2()` e `fat3()` em um outro programa **Python**?

```
1  def fat1(n):
2    res = 1
3    while n > 1:
4      res *= n
5    return res
6  ##
7  def fat2(n):
8    res = 1
9    for i in range(1,n+1):
10     res *= i
11   return res
12 ##
13 def fat3(n):
14   if n == 1: return 1
15   if n > 1: return fat3(n-1) * n
16 ##
17 fat = fat1
18 ##
```

Questão 3

Considere que o código abaixo é uma continuação da questão anterior.
O que acontece com o programa abaixo quando o executamos direto do terminal (como um script)?
E se ele for importado como um módulo?

```
if __name__ == "__main__":
  import sys
  if(len(sys.argv)<2):
    print("Modo de usar: {} {} n".format("python3", sys.argv[0]))
  else:
    print(fat(int(sys.argv[1])))
    print(res)
else:
  print("Importando módulo...")
```

Questão 4

Considere o seguinte código armazenado dentro do arquivo trataString.py:

```
def fazAlgo(string):
  pos = len(string)-1
  stringMi = string.lower()
  string = string.upper()
  stringRe = ""
  vogais = "AEIOU"
  while pos >= 0:
    if string[pos] in vogais:
      stringRe = stringRe + string[pos]
    else:
      stringRe = stringRe + stringMi[pos]
    pos = pos - 1
  return stringRe

##

if __name__ == "__main__":
  print(fazAlgo("teste"))
  print(fazAlgo("o ovo do avestruz"))
  print(fazAlgo("A CASA MUITO ENGRAÇADA"))
  print(fazAlgo("A TELEvisão queBROU"))
  print(fazAlgo("A Vaca Amarela"))
```

O que acontece se o codigo for executado como um script? E se ele for importado como um módulo?

Questão 5

O que o programa abaixo imprime?

```
## classe vazia
class Carro:
  pass
##
meu_carro  = Carro() #criamos um objeto do tipo carro (instância)
type(meu_carro)
## atributos
meu_carro.ano = 2010
meu_carro.modelo = "Uno"
meu_carro.cor = "azul"
##
print(meu_carro)
##
print(meu_carro.ano)
print(meu_carro.modelo)
print(meu_carro.cor)
##
##
outro_carro = Carro() #outro objeto
outro_carro.ano = 1999
outro_carro.modelo = "Gol"
##
print(outro_carro)
##
print(outro_carro.ano)
print(outro_carro.modelo)
print(outro_carro.cor)
##
##
terceiro_carro = meu_carro
terceiro_carro.cor = "amarelo"
terceiro_carro.ano += 1
##
print(terceiro_carro)
##
print(terceiro_carro.ano)
print(terceiro_carro.modelo)
print(terceiro_carro.cor)
##
print(meu_carro.ano)
print(meu_carro.cor)
```

Questão 6

O que o programa abaixo imprime?

```
## classe vazia
class Pato:
  pass
##
pato = Pato()
patinho = Pato()
##
if pato == patinho:
  print("Estamos no mesmo endereço!")
else:
  print("Estamos em endereços diferentes!")
##
print(id(pato))
print(id(patinho))
```

Questão 7

O que o programa abaixo imprime?

```
## classe
class Cafeteira:
  def __init__(self, marca, tipo, tamanho, cor):
    self.marca = marca
    self.tipo = tipo
    self.tamanho = tamanho
    self.cor = cor
##
#novo objeto
minha_cafeteira = Cafeteira('Bialleti', 'moka', 10, 'preta')
##
print(minha_cafeteira.modelo)
print(minha_cafeteira.tipo)
print(minha_cafeteira.tamanho)
print(minha_cafeteira.cor)
##
minha_cafeteira.tamanho = 20
minha_cafeteira.cor = "vermelho"
##
print(minha_cafeteira.tamanho)
print(minha_cafeteira.cor)
##
print(id(minha_cafeteira)
```

Questão 8

Considere o programa abaixo:

```
## classe
class Cachorro:
  def __init__(self, raça, idade, nome, cor):
    self.raça = raça
    self.idade = idade
    self.nome = nome
    self.cor = cor

rex = Cachorro('vira-lata', 2, 'Bobby', 'marrom')
```

Assinale as alternativas que tem como resultado um booleano `True`:

(a) `Bobby.cor == "marrom"`

(b) `rex.idade == "2"`

(c) `rex.nome == "rex"`

(d) `rex.idade > 2`

(e) `"vira-lata"  == rex.raça`

Questão 9

O que o programa abaixo imprime?

```
## classe
class Carro:
  def __init__(self, modelo, ano, cor, velocidadeMax):
    self.modelo = modelo
    self.ano    = ano
    self.cor    = cor
    self.velocidade = 0 # todos os carros começam parados
    self.velocidadeMax = velocidadeMax

  def imprima(self):
    if self.velocidade == 0:
      print("Carro parado!")
    elif self.velocidade < self.velocidadeMax:
      print("{} {} indo a {} km/h".format(self.modelo,
                                          self.cor, self.velocidade))
    else:
      print("{} indo muuuuito rápido!!".format(self.modelo))

  def acelera(self, valor):
    self.velocidade += valor
    self.imprima() # um método pode chamar outro método
```

```
22
23    def pare(self):
24        self.velocidade = 0
25        self.imprima()
26
27  ##
28  carro = Carro('Fusca', 1980, 'preto', 80) #novo objeto
29  ##
30  carro.acelera(40)
31  carro.acelera(20)
32  carro.acelera(50)
33  carro.pare()
34  carro.acelera(100)
35  ##
36  carro.pare()
37  carro.velocidadeMax = 100
38  carro.acelera(80)
39  carro.acelera(1)
40  carro.acelera(19)
41  carro.acelera(1)
42  ##
```

Questão 10

Defina a classe `Triangulo` cujo construtor recebe 3 valores inteiros correspondentes aos lados a, b e c de um triângulo.

A classe triângulo também deve possuir um **método** `perimetro()`, que não recebe parâmetros e devolve um valor inteiro correspondente ao perímetro do triângulo.

```
1  t = Triangulo(1, 1, 1)
2  # deve atribuir uma referência para um triângulo de lados 1, 1 e 1
3  # à variável t
```

Um objeto desta classe deve responder às seguintes chamadas:

```
1  t.a
2  # deve devolver o valor do lado a do triângulo
3  t.b
4  # deve devolver o valor do lado b do triângulo
5  t.c
6  # deve devolver o valor do lado c do triângulo
7  t.perimetro()
8  # deve devolver um inteiro correspondente ao valor do perímetro
9  # do triângulo.
```

Além disso, escreva o metodo `tipo_lado()` que devolve uma string dizendo se o triângulo é:

- isósceles (dois lados iguais)
- equilátero (todos os lados iguais)
- escaleno (todos os lados diferentes)

Note que se o triângulo for equilátero, a função não deve devolver isóceles.

Exemplos:

```
t = Triangulo(4, 4, 4)
t.tipo_lado()
# deve devolver 'equilátero'
u = Triangulo(3, 4, 5)
t.tipo_lado()
# deve devolver 'escaleno'
w = Triangulo(10, 10, 12)
w.tipo_lado()
# deve devolver 'isósceles'
```