UNIVERSIDADE FEDERAL DO CEARÁ – UFC SOBRAL
TÉCNICAS DE PROGRAMAÇÃO – 2019.1 – PROF. WENDLEY

AULA PRÁTICA: 08
03/06/2019

Math e Classes em Python

www.ec.ufc.br/wendley

Para realizar os exercícios práticos adiante, utilize a linguagem Python.

PARTE 1 – Importando módulos

Digite no *prompt* do IDLE os seguintes comandos:

```
>>> sqrt(9)
```

Você deve ter ter como resultado uma mensagem de erro. Isso acontece pois devemos importar o módulo Math antes de usar a função sqrt.

Em seguida, digite:

```
>>> from math import sqrt
>>> sqrt(9)
```

Em seguida, digite:

```
>>> from math import sqrt as raiz
>>> raiz(9)
```

Em seguida, digite `del sqrt` e faça os procedimentos necessários para calcular o seno de 90 graus. Dica: use `math.sin(math.radians(90))`

Exercício 1:
Explore e use 3 métodos dentro de math, diferentes dos usados nos exemplos. Dica: digite no IDLE `dir(math)`.

PARTE 2 – Classes

Na sua sintaxe mais elementar definimos uma classe conforme abaixo:

```
class NomeDaClasse(object):
        pass
```

E um método (função) como:

```
def metodo(args):
        pass
```

onde *args* são argumentos opcionais (parâmetros de entrada). A função *metodo* pode retornar um valor de saída:

```
def metodo(args):
        return args
```

Juntando os dois, temos:

```
class NomeDaClasse(object):
        atributo1 = None

        def metodo(self, args):
                pass
```

A primeira pergunta que você vai ter é o porque do *self* em *metodo*. Uma resposta curta é, todo método criado dentro de uma classe deve definir como primeiro parâmetro o self. A segunda pergunta é: para que serve o *pass*?

A resposta é que, em Python, ao contrário de várias outras linguagens de programação, os blocos de código NÃO são definidos com os caracteres { e }, mas sim com indentação e o caractere :. Devido a esse fato, Python necessita de algo para explicitar quando se quer definir um bloco vazio. O *pass* foi criado exatamente para explicitar essa situação.

Exemplo: Calculadora

Em um novo arquivo (faça CTRL+N), digite o código abaixo e salve em um arquivo chamado `calculadora.py`.

```
#calculadora.py
class Calculadora(object):

    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
```

Em seguida, compile (use a Tecla F5) . Não deve acontecer nada na tela.

No IDLE (modo interativo), digite:

```
>>> from calculadora import Calculadora
>>> obj = Calculadora(128,2)
>>> print('Soma:', obj.soma())
>>> print('Subtração:', obj.subtrai())
>>> print('Multiplicação:', obj.multiplica())
>>> print('Divisão:', obj.divide())
```

Herança

A classe Pais abaixo herda as características da classe Pessoas.

```
class Pessoa(object):
    FEMALE = 0
    MALE = 1

    def __init__(self, nome, sexo):
        super(Pessoa, self).__init__()
        self.nome = nome
        self.sexo = sexo

    def __str__(self):
        return str(self.nome)

class Pais(Pessoa):

    def __init__(self, nome, sexo, crianca):
        super(Pais, self).__init__(nome, sexo)
        self.crianca = criança #nome da criança

    def getCrianca(self):
        return self.crianca

    def __str__(self):
        pass
```

Exercício 2:

Escreva, em um outro arquivo, um algoritmo para instanciar dois objetos para cada um dos tipos, dois para a classe Pessoa e outros dois para Pais, e atribua todos os valores possíveis para os respectivos objetos.

Dica: para os objetos do tipo Pessoa, faça:

```
from _________ import _________
obj1 = Pessoa(“Fulano”,1)
obj2 = _________ (“Fulana”,0)
print obj1._________
print obj2._________
```

O processo é semelhante para a criação do objeto para a classe Pais. Use o método getCrianca() disponível na classe Pais.