# MC-102 — Aula 19
# Funções e Módulos

Prof. Luiz F. Bittencourt

Turmas QR

Instituto de Computação – Unicamp

2019

# Módulos

- Até agora vimos programas cujas funções ficam em um arquivo único, o que é aceitável para programas pequenos.
- É desejável que programas maiores sejam divididos em mais de um arquivo.
  - Se muitas pessoas trabalham em um programa/projeto, é inviável utilizar apenas um arquivo.
- Em Python, módulos permitem a criação de programas em múltiplos arquivos.

# Módulos

- Um módulo é um arquivo .py contendo definições e comandos.
- Exemplo: um módulo circulo.py:

```
pi = 3.14159
def area(raio):
        return pi*(raio**2)

def circunferencia(raio):
        return 2*pi*raio

def superficieEsfera(raio):
        return 4.0*area(raio)

def volumeEsfera(raio):
        return (4.0/3.0)*pi*(raio**3)
```

# Módulos

- Um programa pode acessar um módulo usando o comando **import**.

```
import circulo
print(circulo.pi)
print(circulo.area(3))
print(circulo.circunferencia(3))
print(circulo.superficieEsfera(3))
```

# Módulos

- Módulos geralmente ficam em arquivos individuais, separando conteúdos que estão em um mesmo contexto.
- Cada módulo tem sua tabela de símbolos privada (isto é, nomes de variáveis, funções, etc).
- Note que dentro do arquivo circulo.py, acessamos seus objetos da maneira usual (pi, area, etc).
- Ao usar import, para referenciar objetos do módulo utilizamos a notação de ponto: circulo.pi, circulo.area(3)...

```
import circulo
pi=3
print(circulo.pi)
print(circulo.area(3))
print(pi)
```

# Módulos

- A forma de utilização com notação de pontos evita que o programador precise conhecer os nomes de todos os objetos em todos os módulos que são utilizados.
- Evita, por exemplo, que uma variável de um módulo tenha seu valor acidentalmente alterado, como no exemplo do **pi** do slide anterior.
- É possível alterar esse comportamento usando **from circulo import ***, fazendo com que a tabela de símbolos seja a mesma.

```
from circulo import *
print(pi)
print(area(3))
```

# Módulos

- É possível criar módulos que também funcionam como um programa independente.
- É fundamental que o próprio código saiba se está sendo usado como um módulo ou sendo executado como um programa.
  - O interpretador Python deve saber se inicia a execução de comandos ou apenas importa as funções para uso.
- Uma variável (na verdade um atributo de classe) chamada **__name__** pode ser utilizada para saber se o código está sendo executado como um programa principal ou usado como módulo.
  - **__name__** assume o valor "__main__" quando o arquivo .py é executado como programa principal,

# Módulos

```
pi = 3.14159

def area(raio):
    return pi*(raio**2)

def circunferencia(raio):
    return 2*pi*raio

def superficieEsfera(raio):
    return 4.0*area(raio)

def volumeEsfera(raio):
    return (4.0/3.0)*pi*(raio**3)

def nome():
    print("Nome: %s" %__name__)

if __name__ == '__main__':
    print("Nome: %s" %__name__)
    print(circunferencia(20))
```