centro universitário FEI

CC8210 – NCA210 Programação Avançada I

Prof. Reinaldo A. C. Bianchi
Prof. Isaac Jesus da Silva
Prof. Danilo H. Perico

Módulos
Modules

Módulos

- Para o Python, ***Módulos*** são arquivos fonte que podem ser importados para um programa.
- Podem conter qualquer estrutura do Python e são executados quando importados.
- Um módulo é um arquivo ***.py*** que pode conter:
  - Funções
  - Variáveis e Constantes

Módulos

- Os módulos são carregados através da instrução ***import***.
- Desta forma, ao usar alguma estrutura do módulo, é necessário identificá-lo.
  - Isto é chamado de ***importação absoluta***.

Módulos

- Exemplo de ***importação absoluta***:

```
1  import os
2
3  os.remove('arquivo.txt')
```

Módulos

- Também é possível importar de forma ***relativa***, utilizando as palavras-chaves ***from*** e ***import***
- Exemplo:

```
1  from os import remove
2
3  remove('arquivo.txt')
```

Módulos

- Ainda usando a ***importação relativa***, podemos usar o * para importar tudo o que está no módulo.
- Exemplo:

```
1  from os import *
2
3  remove('arquivo.txt')
```

Módulos

- A ***importação relativa*** com * não é recomendada, pois pode gerar problemas, como a ofuscação de variáveis.

Módulos - Vantagens

- Deixa o programa mais **organizado**
- Permite a **reusabilidade**
- A criação de módulos com funções também **evita** a criação de **variáveis globais**
    - A criação de variáveis globais é, muitas vezes, vista como uma má prática de programação

Criando seus próprios Módulos

Módulo: arquivo *“calculo.py”*

```python
# arquivo calculo.py
# modulo para cálculo de media

#função media recebe lista como parâmetro
def media(lista):
    return float( sum(lista) / len(lista) )
```

Importação absoluta

```python
import calculo

l = [10, 15, 87, 14, 98, 41, 30, 36]

print(calculo.media(l))
```

Importação relativa

```python
from calculo import media

l = [10, 15, 87, 14, 98, 41, 30, 36]

print(media(l))
```

Criando seus próprios Módulos

Módulo: arquivo *“calculo.py”*

```python
# arquivo calculo.py
# modulo para cálculo de media

#função media recebe lista como parâmetro
def media(lista):
    return float( sum(lista) / len(lista) )
```

Importação absoluta renomeando o módulo

```python
import calculo as calc

l = [10, 15, 87, 14, 98, 41, 30, 36]

print(calc.media(l))
```

Importação relativa renomeando a função

```python
from calculo import media as mean

l = [10, 15, 87, 14, 98, 41, 30, 36]

print(mean(l))
```

Módulos

- Módulos são arquivos ***.py***
- Como garantimos que um programa principal não será importado em outro arquivo ***.py*** ?
- Utilizamos a variável especial ***__name__***
- ***__name__*** indica o nome do contexto em que o módulo está sendo executado

Módulos

- Quando um módulo é diretamente executado, ***__name__*** é definido como ***“__main__”***
- Quando o módulo é importado, ***__name__*** fica igual ao nome do módulo, ou seja, do arquivo.

Módulos

- Logo, podemos incluir uma condição para rodar nosso ***main***:

```
1  from calculo import media
2
3
4  def main():
5      l = [10, 15, 87, 14, 98, 41, 30, 36]
6      print(media(l))
7
8  if __name__ == "__main__":
9      main()
```

Biblioteca padrão

- O Python vem com **vários módulos distribuídos por padrão** com o interpretador

Biblioteca padrão

- Alguns módulos importantes da biblioteca padrão são:
  - Matemática: *math*, *random*
  - Sistema: *os*, *sys*, *zipfile*
  - Tempo: *time*, *datetime*

Pacotes
Packages

Pacotes

- De uma forma simples, um ***pacote*** é uma coleção de módulos.
- Quando os módulos aumentam de tamanho, podemos dividí-los em pacotes
- ***Módulos*** são estruturados em ***arquivos***, enquanto que, ***pacotes*** são estruturados em ***pastas***.
- Todos os ***pacotes*** devem conter um arquivo ***__init__.py***
- Os arquivos ***__init__.py*** servem para que o interpretador possa identificar quais diretórios são pacotes e quais não são.
- ***__init__.py*** pode ser apenas um arquivo vazio, mas pode também executar código de inicialização do pacote.

Pacotes

- Para acelerar o carregamento de módulos, o Python guarda versões compiladas de cada módulo no diretório ***__pycache__***

- Em ***__pycache__*** serão criados arquivos com a extensão ***.pyc***, esses arquivos são versões compiladas por bytecode.

- Para seu programa você pode ignorar a existência desses arquivos, mas eles tornarão seu programa um pouco mais rápido.

Exercícios
Módulos

Exercícios

Em todos os exercícios certifique-se de que o programa principal não será executado se o arquivo que contém sua solução for importado para outro programa.

Exercícios

1. Neste exercício, você deve programar um módulo composto por 4 funções que, juntas, servirão para determinar se uma senha é boa ou não. Uma boa senha deve ter:
    a. Pelo menos 8 caracteres (1ª função)
    b. Pelo menos uma letra maiúscula (2ª função)
    c. Pelo menos uma letra minúscula (3ª função)
    d. Pelo menos um número (4ª função)

Exercícios

1. Cada função deve retornar *true* ou *false* para a senha recebida.

   Faça também um programa principal que leia uma senha do usuário, chame cada função de verificação e relate se a senha é boa ou não.

Exercícios

2. Crie um módulo com 3 funções que, juntas, servirão para a criação de placas de veículos no novo padrão do Mercosul. O novo padrão é composto por quatro letras e três números: AAA0A00

   Assim, sua primeira função deve gerar 4 letras aleatoriamente, a segunda função deve gerar 3 números inteiros aleatoriamente e a terceira função deve juntar as letras e os números no padrão correto: letra-letra-letra-num-letra-num-num (AAA0A00).

Exercícios

2. Escreva um programa principal que chama as três funções e exibe a placa criada.

Exercícios

3. Escreva um módulo que contenha uma função que receba os comprimentos dos dois lados mais curtos de um triângulo retângulo (catetos) como seus parâmetros. A função deve retornar a hipotenusa do triângulo, calculada usando o teorema de Pitágoras.

Escreva também o programa principal que recebe do usuário os comprimentos dos catetos do triângulo retângulo, chama a função para calcular a hipotenusa e exibe o resultado.

Exercícios

4. Muitas pessoas não usam letras maiúsculas corretamente, especialmente ao digitar em pequenos dispositivos como smartphones. Neste exercício, você escreverá um módulo composto por uma função que capitaliza os caracteres apropriados em uma *string*. O primeiro caractere da *string* deve ser sempre capitalizado, assim como o primeiro caractere após “.”, “!” ou “?”.