Aula 01

Módulos

Em Python, um módulo é um arquivo contendo código Python que pode ser reutilizado em outros programas. Os módulos são uma forma de organizar o código em partes separadas e reutilizáveis, o que torna o desenvolvimento de software mais fácil e eficiente.

Para usar um módulo em um programa Python, primeiro é necessário importá-lo. O comando "import" é usado para importar um módulo em um programa Python. Por exemplo, para importar o módulo "math", que contém funções matemáticas comuns, podemos usar o seguinte comando:

```
>>> import math
```

Isso torna as funções e variáveis do módulo math disponíveis em seu programa.

Por exemplo, para usarmos a função sqrt (raiz quadrada) do módulo math, podemos escrever:

```
>>> import math
>>>
>>> print(math.sqrt(25))
5.0
>>>
```

O código acima vai imprimir 5.0 no terminal, pois é a raiz quadrada de 25.

Além do comando import, também é possível usar o comando from para importar funções específicas de um módulo. Por exemplo, para importar apenas a função sqrt do módulo math, podemos escrever:

```
>>> from math import sqrt
>>>
>>> print(sqrt(25))
5.0
>>>
```

Isso fará exatamente o que foi feito no exemplo anterior, mas agora sem chamar explicitamente o módulo math.

Módulo sys

O módulo sys é um módulo integrado do Python que fornece acesso a algumas variáveis e funções relacionadas ao sistema. Ele permite que um programador interaja com o interpretador do Python em tempo de execução, obtenha informações sobre o ambiente em que o código está sendo executado e controle o comportamento do programa.

Aqui estão algumas das principais funções e variáveis disponíveis no módulo sys.

Funções:

sys.argv

A variável sys.argv contém uma lista de argumentos de linha de comando passados para o programa. O primeiro elemento da lista sys.argv[0] é o nome do programa em si.

Por exemplo, se o seguinte comando for executado a partir do terminal: python myprogram.py arg1 arg2, o valor de sys.argv será [myprogram.py, arg1, arg2].

sys.exit

A função sys.exit() é usada para encerrar o programa imediatamente. Um argumento inteiro opcional pode ser fornecido para indicar o status de saída do programa. Um valor de 0 indica sucesso, enquanto valores maiores que 0 indicam algum tipo de erro.

Por exemplo, se o programa precisar sair prematuramente devido a um erro, pode-se usar sys.exit(1) para indicar que ocorreu um erro.

Variáveis:

sys.path

A variável sys.path é uma lista de diretórios onde o interpretador Python procura por módulos importados. É possível adicionar novos diretórios à lista para permitir que o interpretador Python encontre módulos personalizados.

Por exemplo, se o módulo mymodule estiver localizado no diretório /path/to/mymodule, o diretório pode ser adicionado ao sys.path usando sys.path.append('/path/to/mymodule').

sys.version

A variável sys.version contém uma string que representa a versão atual do interpretador Python.

Por exemplo, a string 3.9.4 (default, Apr 5 2021, 01:03:40) [GCC 10.2.1 20201125 (Red Hat 10.2.1-9)] indica que o interpretador Python está na versão 3.9.4 e foi compilado com GCC 10.2.1 em 5 de abril de 2021.

sys.platform

A variável sys.platform contém uma string que representa o sistema operacional em que o interpretador Python está sendo executado.

Por exemplo, a string win32 indica que o interpretador Python está sendo executado no Microsoft Windows. Outros valores possíveis incluem linux, darwin (macOS) e cygwin.

Função dir

A função dir() do Python é uma função integrada (built-in) que permite listar os atributos (métodos e propriedades) de um objeto. Essa função é frequentemente usada para inspecionar objetos em tempo de execução e explorar suas funcionalidades.

A sintaxe da função dir() é simples: basta chamar a função passando o objeto como argumento. Por exemplo, para listar os atributos de uma lista em Python, pode-se usar o seguinte código:

```
>>> lista = [1, 2, 3]
>>> print(dir(lista))
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '
__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getat
tribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '
__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '
__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__rep
r__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', '
index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

Ao executar esse código, será exibida uma lista de strings contendo os atributos disponíveis para a lista, como append, clear, count, extend, index, insert, pop, remove, reverse e sort. Cada um desses atributos é um método ou uma propriedade da lista que pode ser acessado e usado para realizar operações na lista.

É importante ressaltar que a função dir() lista todos os atributos disponíveis para um objeto, incluindo aqueles que são específicos da classe do objeto e aqueles que são herdados de superclasses. Além disso, a função dir() também lista alguns atributos especiais que começam e terminam com __, como __class__, __doc__, __str__ e outros.

Exercícios para Praticar

1. Como importar o módulo sys em um arquivo Python?
2. O que o método sys.argv retorna?
3. Como imprimir o caminho completo do executável Python em um script usando sys?
4. Como imprimir a versão do Python em execução usando sys?
5. Como obter a quantidade de referências em uso no momento usando sys?
6. Como obter o tamanho máximo permitido para objetos Python usando sys?
7. Como alterar o tamanho máximo permitido para objetos Python usando sys?
8. Como forçar a saída de um programa Python usando sys?
9. Como redirecionar a saída de um programa Python para um arquivo usando sys?
10. Como redirecionar a entrada de um programa Python para um arquivo usando sys?
11. Como obter informações sobre o sistema operacional usando sys?
12. Como obter informações sobre a plataforma Python em execução usando sys?
13. Como obter o diretório atual de trabalho usando sys?
14. Como adicionar um caminho à lista de caminhos de pesquisa do interpretador Python usando sys?
15. Como remover um caminho da lista de caminhos de pesquisa do interpretador Python usando sys?
16. Como obter informações sobre a codificação padrão usada pelo interpretador Python usando sys?
17. Como definir a codificação padrão usada pelo interpretador Python usando sys?
18. Como obter o tamanho do buffer de saída padrão usado pelo interpretador Python usando sys?
19. Como definir o tamanho do buffer de saída padrão usado pelo interpretador Python usando sys?
20. Como obter informações sobre a arquitetura da máquina em execução usando sys?
21. Crie um programa que utilize a função dir() para listar os atributos disponíveis para um objeto do tipo str.
22. Escreva um código que utilize a função dir() para listar os métodos disponíveis para um objeto do tipo int.
23. Faça um programa que liste os atributos de uma lista vazia utilizando a função dir().
24. Crie um código que liste os atributos de um dicionário utilizando a função dir().
25. Escreva um programa que utilize a função dir() para listar os atributos de um objeto do tipo tuple.
26. Faça um código que liste os atributos de um objeto do tipo set utilizando a função dir().
27. Crie um programa que utilize a função dir() para listar os métodos disponíveis para um objeto do tipo float.
28. Escreva um código que liste os atributos de um objeto do tipo bool utilizando a função dir().
29. Faça um programa que liste os atributos de uma função em Python utilizando a função dir().
30. Crie um código que utilize a função dir() para listar os atributos disponíveis para o módulo math em Python.