Aula 01

Revisão

Formas de realizar a importação de um módulo em Python

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

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

Módulo sys é usado para termos acesso a algumas variáveis e funções relacionadas ao sistema.

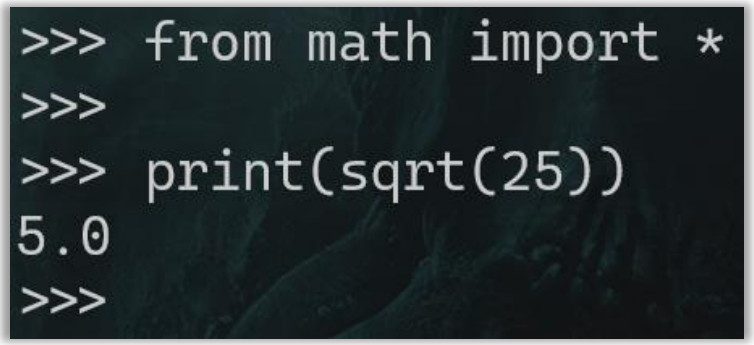
```
>>> import sys
>>> print(sys.argv)
['']
>>> print(sys.version)
3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39
) [MSC v.1934 64 bit (AMD64)]
>>> print(sys.platform)
win32
>>> print(sys.path)
['', 'C:\\Python\\Python311\\python311.zip', 'C:\\P
ython\\Python311\\Lib', 'C:\\Python\\Python311\\DLL
s', 'C:\\Users\\augusto.hertzog\\AppData\\Roaming\\
Python\\Python311\\site-packages', 'C:\\Users\\augu
sto.hertzog\\AppData\\Roaming\\Python\\Python311\\s
ite-packages\\win32', 'C:\\Users\\augusto.hertzog\\
AppData\\Roaming\\Python\\Python311\\site-packages\
\win32\\lib', 'C:\\Users\\augusto.hertzog\\AppData\
\Roaming\\Python\\Python311\\site-packages\\Pythonw
in', 'C:\\Python\\Python311', 'C:\\Python\\Python31
1\\Lib\\site-packages']
>>>
>>> sys.exit()
PS C:\Users\augusto.hertzog>
```

Importações com Wildcard

A importação em Python é usada para incluir um módulo em um programa Python, permitindo que você acesse as classes, funções e variáveis definidas no módulo importado.

Ao importar um módulo em Python usando o comando import, você pode especificar o nome do módulo que deseja importar. Por exemplo, para importar o módulo math, você pode escrever "import math".

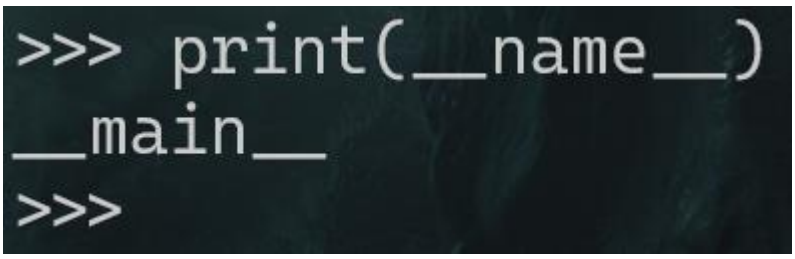
No entanto, quando você usa um wildcard (*) na importação, você importa todos os símbolos públicos (classes, funções e variáveis) definidos no módulo. Por exemplo, se você quiser importar todos os símbolos públicos do módulo math, você pode escrever from "math import *".



O uso de wildcard na importação pode facilitar o acesso a todos os símbolos públicos de um módulo, pois não é necessário especificar cada um deles individualmente. No entanto, é importante ter cuidado ao usar um wildcard na importação, pois isso pode tornar seu código mais difícil de entender e manter. Isso ocorre porque você pode acabar importando símbolos que não são necessários para o seu código, ou pode haver conflitos de nome entre os símbolos importados e os símbolos definidos no seu código.

__name__ == '__main__'

Em Python, o módulo especial "__name__" é uma variável interna que representa o nome do módulo atual (ela possui 2 sublinhados antes e depois). Quando um arquivo Python é executado, o valor de "__name__" para esse arquivo será definido como '__main__'.



A expressão "__name__ == '__main__'" é comumente usada em arquivos Python para testar se o arquivo está sendo executado como um programa principal ou se está sendo importado como um módulo por outro arquivo. Essa expressão é geralmente colocada no final do arquivo Python, após a definição de funções e outras classes.

Se o arquivo Python estiver sendo executado como um programa principal, o valor de __name__ será '__main__', e a expressão __name__ == '__main__' será avaliada como True. Nesse caso, o código dentro do bloco if __name__ == '__main__': será executado. Esse bloco é comumente usado para incluir o código que é executado quando o arquivo é executado como um programa principal.

```python
"""módulo de números de Fibonacci"""


def escreve_fib(n):
    """escreve a série de Fibonacci até n"""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()


def retorna_fib(n):
    """retorna a série de Fibonacci até n"""
    resultado = []
    a, b = 0, 1
    while a < n:
        resultado.append(a)
        a, b = b, a+b
    return resultado


if __name__ == '__main__':
    # inicia o programa
    import sys
    escreve_fib(int(sys.argv[1]))

print(f'Valor de __name__ : {__name__}.')

```

```
C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>python fibonacci.p
y 100
0 1 1 2 3 5 8 13 21 34 55 89
Valor de __name__ : __main__.

C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
```

Acima, temos duas implementações da função de Fibonacci, que vai retornar ou mostrar todos os valores até n e o valor atual de __name__. Repare que ao ser chamado por outro módulo, ele vai ter seu valor alterado.

```python
import fibonacci

lista = fibonacci.retorna_fib(100)

print(lista)

```

```
C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>python main.py
Valor de __name__ : fibonacci.
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
```

se o arquivo Python estiver sendo importado como um módulo por outro arquivo, o valor de __name__ será o nome do módulo e a expressão __name__ == '__main__' será avaliada como False. Nesse caso, o código dentro do bloco if __name__ == '__main__': não será executado, pois esse bloco só é executado quando o arquivo é executado como um programa principal.

O uso da expressão __name__ == '__main__' é útil porque permite que o mesmo arquivo Python seja usado tanto como um programa principal quanto como um módulo. Quando o arquivo é importado como um módulo, o código dentro do bloco if __name__ == '__main__': não é executado, mas as funções e outras classes definidas no arquivo ainda podem ser usadas pelo código que o importa.

Isso é muito útil quando queremos desenvolver diversas funcionalidades, mas não queremos ficar importando para outro módulo. Assim podemos testar todas as funcionalidades antes de utilizar no nosso programa principal.

Se adicionarmos um comentário com as aspas triplas na primeira linha do módulo, ela vai aparecer como documentação dele.

Arquivos

Em Python, arquivos são usados para armazenar informações permanentemente em disco. Existem três tipos principais de arquivos: arquivos de texto, arquivos binários e arquivos CSV.

Para trabalhar com arquivos em Python, você deve seguir os seguintes passos:

1. Abrir o arquivo: Você deve usar a função "open()" para abrir o arquivo. O primeiro parâmetro da função é o nome do arquivo e o segundo parâmetro é o modo de abertura do arquivo. Existem diferentes modos de abertura de arquivo, que incluem:
    - "r" (leitura): abre o arquivo para leitura;
    - "w" (escrita): abre o arquivo para escrita, apagando o conteúdo anterior, se houver;
    - "a" (adição): abre o arquivo para adição de conteúdo no final do arquivo;
    - "x" (criação): cria um novo arquivo para escrita;

2. Ler ou escrever no arquivo: Depois de abrir o arquivo, você pode ler ou escrever nele, dependendo do modo em que ele foi aberto. Para escrever no arquivo, você deve usar a função "write()". Para ler o conteúdo do arquivo, você pode usar a função "read()".

3. Fechar o arquivo: É importante sempre fechar o arquivo após a leitura ou escrita, usando a função close(). Isso garante que todos os dados escritos no arquivo sejam armazenados e que o arquivo não seja corrompido.

Veja o exemplo abaixo de leitura:

```python
"""Documentação do módulo"""

def abre_arquivo():
    """abre um arquivo e mostra o conteúdo"""
    arq = open('um_arquivo.txt', 'r')

    conteudo = arq.read()

    print(conteudo)

    arq.close()

if __name__ == '__main__':
    abre_arquivo()

```

```
PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
 python .\main.py
o conteÃºdo do arquivo

PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
```

Exemplo de escrita:

```python
"""Documentação do módulo"""

def abre_arquivo():
    """abre um arquivo e mostra o conteúdo"""
    arq = open('um_arquivo.txt', 'r')
    conteudo = arq.read()
    print(conteudo)
    arq.close()

def escreve_arquivo():
    """função para escrever um texto em um arquivo."""
    arquivo = open('um_arquivo.txt', 'w')

    arquivo.write('isso vai apagar o conteúdo original')

    arquivo.close()

if __name__ == '__main__':
    abre_arquivo()
    escreve_arquivo()
    abre_arquivo()

```

```
PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
 python .\main.py
o conteÃ°do do arquivo

isso vai apagar o conteúdo original
PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
```

Lembre-se de que, ao abrir um arquivo para escrita, todo o conteúdo anterior será apagado. Se você deseja adicionar conteúdo ao arquivo sem apagar o conteúdo anterior, deve abrir o arquivo no modo de adição ("a"). Adicionando um conteúdo no final do arquivo.

```python
"""Documentação do módulo"""

def abre_arquivo():
    """abre um arquivo e mostra o conteúdo"""
    arq = open('um_arquivo.txt', 'r')
    conteudo = arq.read()
    print(conteudo)
    arq.close()

def escreve_arquivo():
    """função para escrever um texto em um arquivo."""
    arquivo = open('um_arquivo.txt', 'w')
    arquivo.write('isso vai apagar o conteúdo original')
    arquivo.close()

def adiciona_arquivo():
    """função que vai adicionar um texto ao final do arquivo."""
    arq = open('um_arquivo.txt', 'a')
    arq.write('isso vai ser adicionado ao final do arquivo.')
    arq.close()

if __name__ == '__main__':
    abre_arquivo()
    escreve_arquivo()
    abre_arquivo()
    adiciona_arquivo()
    abre_arquivo()

```

```
PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
 python .\main.py
isso vai apagar o conteúdo original
isso vai apagar o conteúdo original
isso vai apagar o conteúdo originalisso vai ser adicionado ao f
inal do arquivo.
PS C:\Users\augusto.hertzog\GitHub\master-python\antigo\python>
```

Assim como as strings, podemos formatar os textos inseridos no arquivo usando os caracteres especiais como \n, \t etc.

Exercícios para Praticar

Exercícios Wildcard

Com wildcard:

1. Importe todas as funções do módulo math e calcule a raiz quadrada de 25.
2. Importe todas as funções do módulo sys e use a função sys.argv para obter argumentos da linha de comando.
3. Importe todas as funções do módulo math e calcule o seno de 45 graus usando a função math.sin().
4. Importe todas as funções do módulo sys e use a função sys.path para obter o caminho de busca de módulos.
5. Importe todas as funções do módulo math e calcule o logaritmo natural de 10 usando a função math.log().
6. Importe todas as funções do módulo sys e use a função sys.stdin.readline() para ler entrada do usuário.
7. Importe todas as funções do módulo math e calcule a tangente de 30 graus usando a função math.tan().
8. Importe todas as funções do módulo sys e use a função sys.platform para obter o nome da plataforma em que o Python está sendo executado.
9. Importe todas as funções do módulo math e calcule a hipotenusa de um triângulo retângulo com catetos de comprimento 3 e 4.
10. Importe todas as funções do módulo sys e use a função sys.exit() para sair do programa com um código de status.

Sem wildcard:

11. Importe a função sqrt do módulo math e calcule a raiz quadrada de 36.
12. Importe a função argv do módulo sys e use-a para obter argumentos da linha de comando.
13. Importe a constante pi do módulo math e use-a para calcular a circunferência de um círculo com raio 2.
14. Importe a função stdout do módulo sys e use-a para imprimir uma mensagem na tela.
15. Importe a função pow do módulo math e use-a para elevar 2 a potência de 3.
16. Importe a função version_info do módulo sys e use-a para obter a versão do Python em que o programa está sendo executado.
17. Importe a função sin do módulo math e use-a para calcular o seno de um ângulo de 60 graus.
18. Importe a função path do módulo sys e use-a para obter o caminho de busca de módulos.
19. Importe a função log10 do módulo math e use-a para calcular o logaritmo de base 10 de 1000.
20. Importe a função exit do módulo sys e use-a para sair do programa com um código de status específico.

Exercícios __name__

1. Crie um módulo Python que imprima "Hello World" quando executado diretamente, mas não faça nada se importado por outro módulo.
2. Crie um módulo Python que implemente uma função que calcule a raiz quadrada de um número e imprima o resultado apenas se o módulo for executado diretamente.
3. Crie um módulo Python que implemente uma função que recebe uma lista de números e imprime a soma dos elementos apenas se o módulo for executado diretamente.
4. Crie um módulo Python que implemente uma função que recebe um número e imprime se ele é par ou ímpar apenas se o módulo for executado diretamente.
5. Crie um módulo Python que implemente uma função que recebe um número e imprime sua tabuada apenas se o módulo for executado diretamente.
6. Crie um módulo Python que implemente uma função que recebe uma lista de palavras e imprime as palavras em ordem alfabética apenas se o módulo for executado diretamente.

Exercícios Arquivos

Conceitos

1. Como abrir um arquivo de texto em modo de leitura em Python?
2. Qual a função usada para ler o conteúdo de um arquivo em Python?
3. Como fechar um arquivo em Python após a leitura?
4. Como abrir um arquivo de texto em modo de escrita em Python?

5. Qual a função usada para escrever em um arquivo em Python?
6. Como fechar um arquivo em Python após a escrita?
7. Como criar um novo arquivo em Python?
8. Qual o modo de abertura de arquivo usado para adicionar conteúdo em um arquivo existente em Python?
9. Como adicionar conteúdo em um arquivo existente em Python?
10. Como verificar se um arquivo existe em Python?
11. Como ler um arquivo linha por linha em Python?
12. Como escrever em um arquivo em Python usando uma lista de strings?
13. Como renomear um arquivo em Python?
14. Como deletar um arquivo em Python?
15. Como abrir um arquivo binário em Python?
16. Qual a função usada para ler o conteúdo de um arquivo binário em Python?
17. Como escrever em um arquivo binário em Python?
18. Qual o modo de abertura de arquivo usado para criar um novo arquivo binário em Python?
19. Como ler e escrever em um arquivo CSV em Python?
20. Qual a biblioteca usada para trabalhar com arquivos CSV em Python?

Práticos

21. Crie um código em Python que leia o conteúdo de um arquivo de texto e o imprima na tela.
22. Crie um código em Python que escreva "Olá, mundo!" em um arquivo de texto.
23. Crie um código em Python que adicione o texto "Isso é incrível!" ao final de um arquivo de texto.
24. Crie um código em Python que crie um novo arquivo de texto chamado "meu_arquivo.txt" e escreva "Este é o meu arquivo de teste." nele.
25. Crie um código em Python que verifique se um arquivo chamado "arquivo_de_teste.txt" existe.
26. Crie um código em Python que leia um arquivo de texto linha por linha e imprima cada linha na tela.
27. Crie um código em Python que escreva as strings "Gato", "Cachorro" e "Passarinho" em um arquivo de texto, cada uma em uma nova linha.
28. Crie um código em Python que adicione as strings "Cavalo", "Porco" e "Vaca" ao final de um arquivo de texto, cada uma em uma nova linha.
29. Crie um código em Python que renomeie um arquivo chamado "arquivo_velho.txt" para "arquivo_novo.txt".
30. Crie um código em Python que delete um arquivo chamado "arquivo_antigo.txt".
31. Crie um código em Python que leia um arquivo de texto e conte o número de palavras nele.
32. Crie um código em Python que leia um arquivo de texto e conte o número de ocorrências de uma palavra específica.
33. Crie um código em Python que leia um arquivo de texto e conte o número de linhas nele.
34. Crie um código em Python que leia um arquivo de texto e imprima apenas as linhas que contêm a palavra "Python".
35. Crie um código em Python que leia um arquivo de texto e imprima apenas as linhas que começam com a letra "A".
36. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as linhas que contêm a palavra "import".
37. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as linhas que contêm um número.
38. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as linhas que contêm uma data no formato "dd/mm/aaaa".
39. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as linhas que contêm um endereço de e-mail válido.
40. Crie um código em Python que leia um arquivo de texto e substitua todas as ocorrências da palavra "bom" por "ótimo".
41. Crie um código em Python que leia um arquivo de texto e remova todas as ocorrências da palavra "mau".
42. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as palavras que têm mais de 10 caracteres.

43. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as palavras que começam com a letra "P".
44. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo apenas com as palavras que terminam com a letra "a".
45. Crie um código em Python que leia um arquivo de texto e conte quantas vezes a palavra "Python" aparece nele.
46. Crie um código em Python que leia um arquivo de texto e conte quantas vezes cada palavra aparece nele.
47. Crie um código em Python que leia um arquivo de texto e imprima a palavra mais comum nele.
48. Crie um código em Python que leia um arquivo de texto e imprima a palavra mais longa nele.
49. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo com as palavras em ordem alfabética.
50. Crie um código em Python que leia um arquivo de texto e crie um novo arquivo com as palavras em ordem decrescente de tamanho.