



# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 07 – MÓDULOS E BIBLIOTECAS

# O QUE IREMOS APRENDER

**01** MÓDULOS

**02** COMO IMPORTAR MÓDULOS NO PYTHON

**03** ATIVIDADES PRÁTICAS

**04** BIBLIOTECAS

**05** ATIVIDADES PRÁTICAS

**06** INTRODUÇÃO AULA 08

# Módulos e Bibliotecas

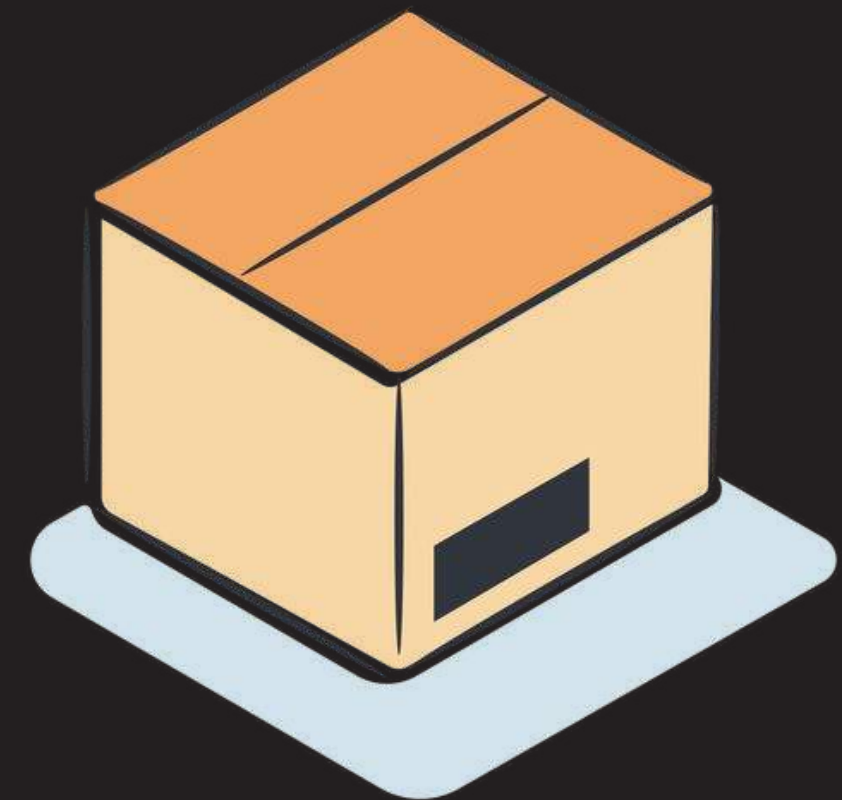
Quando estamos programando, independentemente da linguagem, nossos programas tendem a ficar cada vez maiores, o que traz diversos problemas, como dificuldade de legibilidade e manutenção. Diante desse cenário, é importante que nosso código esteja dividido em diferentes arquivos .py. Em vez de um único arquivo com muitas linhas de código, teremos vários arquivos, cada um com um trecho de código.

Além da importância dos **módulos** para a divisão do código, também precisamos das **bibliotecas** para utilizar funcionalidades prontas e desenvolver de forma mais eficiente. E são esses assuntos que serão abordados nessa aula.

# O que são Módulos

Ao programar, é importante dividir o código em diferentes arquivos `.py`, ou módulos, para evitar problemas como dificuldade de legibilidade e manutenção. Isso permite que cada arquivo contenha um pedaço de código, tornando-o mais organizado.

- Um módulo é um arquivo Python contendo funções, classes e variáveis.
- Você pode criar seus próprios módulos escrevendo código Python em um arquivo com extensão `.py`.
- Para usar um módulo em outro programa, você importa-o usando a instrução `import`.



# O que são Módulos

Criamos uma função chamada **saudacao** que possui o parâmetro **nome**, no arquivo **meu\_modulo.py**



```
1  # meu_modulo.py
2  def saudacao(nome):
3      return f"Olá, {nome}"
```



```
1  import meu_modulo
2  mensagem = meu_modulo.saudacao("Alice")
3  print(mensagem) # Saída: "Olá, Alice!"
```

No arquivo **meu\_arquivo** importamos o **meu\_modulo**. Criamos uma variável **mensagem** que armazena a chamada do arquivo **meu\_modulo** e busca a função **saudacao** que espera um argumento como resposta.

# Como importar Módulos no Python

**01** `import modulo`

**02** `from modulo import função`

**03** `from modulo import (func1, func2)`

**04** `from modulo import *`


**05** `from pacote.modulo import função`

**06** `import modulo as apelido`

# Como importar Módulos no Python

## **import modulo**

Uma advertência, importar um módulo desta maneira traz pra gente TODAS AS FUNÇÕES daquele módulo, ou seja, se neste módulo existirem 100 funções, essa maneira de import traz todas as 100 funções.



```
1 import contar
2
3 contar.caracteres("infinity")
```

## **from modulo import função**

Realizando o import desta maneira temos o cuidado de importar apenas a função que queremos, evitando importar todas as funções daquele módulo.



```
1 from contar import caracteres
2
3 caracteres("infinity")
```



# Como importar Módulos no Python

## **from modulo import (func1, func2)**

Quando queremos ou precisamos importar mais de uma função, devemos importar todas as funções dentro de uma tupla, seguindo o padrão de estilização de código do Python – PEP 8.



```
1 from contar import (  
2     caracteres,  
3     numeros  
4 )  
5  
6 caracteres("teste") # Saída: 5  
7 numeros(2) # Saída: 2**2=4
```

## **from modulo import \***

Nesta maneira importamos tudo o que existe no módulo, basicamente como a primeira maneira.



```
1 from contar import *  
2  
3 numeros(2) # Saída: 2**2=4
```



# Como importar Módulos no Python

## **from pacote.modulo import função**

Dessa maneira podemos importar uma função de um módulo que está em um pacote externo, ou seja, quando o módulo e o arquivo onde o módulo será importado não estão na mesma pasta



```
1  from teste.contar import *  
2  
3  caracteres("Raama")
```

## **import modulo as apelido**

Em algum momento podemos lidar com nomes de módulos grandes, para não termos que digitar nomes grandes todas as vezes que vamos utilizar alguma função daquele módulo, damos um apelido (alias) para este módulo, através da palavra **as**



```
1  import contar as char  
2  
3  char.caracteres("Raama")
```

# Como importar Módulos no Python

## Arquivo: contar.py



```
1 def somarNumeros(num1, num2):  
2     return f"A soma é : {num1 + num2}"
```

Nesse arquivo foi construído apenas a função que será responsável por somar num1 e num2.

## Arquivo: funcao.py



```
1 import contar  
2  
3 numero1 = int(input("Digite o primeiro número: "))  
4 numero2 = int(input("Digite o segundo número: "))  
5  
6 soma = contar.somarNumeros(numero1, numero2)  
7 print(soma)
```

Neste arquivo, o arquivo "contar" foi importado, os números foram solicitados e posteriormente armazenados na variável "soma" e colocados na função "somarNumeros".

# ATIVIDADE PRÁTICA 1

Crie um programa que será uma calculadora. Nesta calculadora você deverá ter um módulo para as operações matemáticas, o arquivo principal deverá conter apenas um menu de escolha para o usuário (soma, subtração, multiplicação e divisão).

# ATIVIDADE PRÁTICA 2

Crie um módulo chamado `manipulacao_strings` que contenha funções para realizar operações com strings, como inverter uma string, contar o número de palavras em uma string e verificar se uma string é um palíndromo (lê-se igual de trás para frente). Crie um programa principal que importe o módulo e use essas funções com strings fornecidas pelo usuário.

# O que são Bibliotecas

As bibliotecas de código são coleções de código predefinido que oferecem funcionalidades específicas e podem ser reutilizadas em vários programas. Elas facilitam o desenvolvimento de software, economizam tempo e evitam a necessidade de reinventar a roda ao realizar tarefas comuns.





# Exemplos de algumas Bibliotecas

O módulo `math` é parte da biblioteca padrão do Python e fornece um conjunto de funções matemáticas e constantes para realizar cálculos matemáticos.

Operações que podem ser realizadas com o `math`

```
1 import math
2
3 num = int(input("Digite um número: "))
4 raiz_quadrada = math.sqrt(num)
5 print(f"A raiz quadrada de {num} é {raiz_quadrada}")
```

Nesse exemplo a biblioteca `math` foi utilizada para encontrar a raiz quadrada de um número

`math.`

- `__annotations__`
- `acos`
- `acosh`
- `asin`
- `asinh`
- `atan`
- `atan2`
- `atanh`
- `cbrt`
- `ceil`
- `comb`
- `copysign`



# Exemplos de algumas Bibliotecas



```
1  import math
2  num = 5.6
3  arredonda_cima = math.ceil(num)
4  arredonda_baixo = math.floor(num)
5  print(f"Arredondando {num} para cima: {arredonda_cima}")
6  print(f"Arredondando {num} para baixo: {arredonda_baixo}")
```

Nesse exemplo foi utilizado o `math.ceil` e o `math.floor` que é utilizado para arredondar o número para baixo ou para cima.

# Exemplos de algumas Bibliotecas

A biblioteca random é usada para gerar números aleatórios em Python. Ela fornece funções para criar números aleatórios, embaralhar listas, escolher elementos aleatórios e muito mais.



```
1 import random
2
3 minha_lista = ["maçã", "banana", "laranja", "uva"]
4 escolha_aleatoria = random.choice(minha_lista)
5 print("Escolha aleatória da lista:", escolha_aleatoria)
```

**Escolhendo um elemento aleatório de uma lista**

random.

- [x] random
- [x] betavariate
- [x] choice
- [x] choices
- [x] expovariate
- [x] gammavariate
- [x] gauss
- [x] getrandbits
- [x] getstate
- [x] lognormvariate
- [x] normalvariate
- [x] paretovariate

# ATIVIDADE PRÁTICA 3

Crie um programa que permite ao usuário calcular a área e perímetro de formas geométricas simples, como quadrados, retângulos e círculos. Use funções matemáticas do módulo `math` para realizar os cálculos.

# ATIVIDADE PRÁTICA 4

Desenvolva um jogo de adivinhação em que o programa escolhe um número aleatório e desafia o jogador a adivinhá-lo. Forneça dicas ao jogador, indicando se o número é maior ou menor do que a adivinhação atual.

# ATIVIDADE PRÁTICA 5

Desenvolva um programa que permita ao usuário converter entre diferentes unidades de medida, como metros para pés, quilogramas para libras, ou Celsius para Fahrenheit. Use módulos que contenham funções de conversão.

# DESAFIO PRÁTICO

## Gerenciador de Livros de Biblioteca - passo 1

Crie um programa que permita aos usuários:

Adicionar novos livros à biblioteca, com informações como título, autor e número de cópias disponíveis.

Listar todos os livros disponíveis na biblioteca.

Permitir aos usuários fazer empréstimos de livros, reduzindo o número de cópias disponíveis quando um livro é emprestado.



# DESAFIO PRÁTICO

## Gerenciador de Livros de Biblioteca - passo 2

Permitir aos usuários devolver livros, aumentando o número de cópias disponíveis quando um livro é devolvido.

Verificar a disponibilidade de um livro específico na biblioteca.

Mostrar a lista de livros emprestados a um usuário específico.



# SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 08 DE PYTHON.  
REVISÃO E FUNÇÕES AGREGADORAS.

The logo consists of the letters 'IN' in a white, bold, sans-serif font, centered within a solid red square.

**INFINITY SCHOOL**

VISUAL ART CREATIVE CENTER



# Funções Agregadoras

As funções agregadoras desempenham um papel fundamental na análise de dados, As funções agregadoras são essenciais na resolução de problemas, permitindo resumir informações, extrair insights e tomar decisões informadas. Elas são ferramentas fundamentais na manipulação de dados e na aplicação de lógica de programação para resolver problemas do mundo real.



# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 07 – MÓDULOS E BIBLIOTECAS