

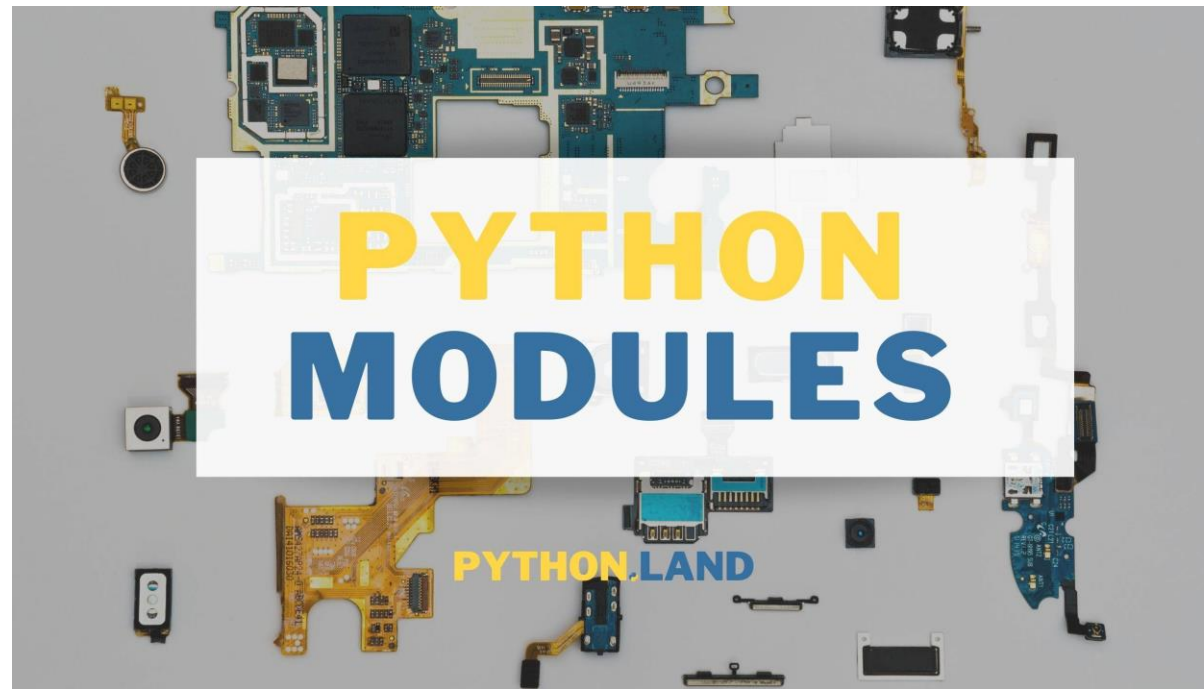
## ALGORITMIA E ESTRUTURAS DE DADOS

### MÓDULO II

#### FUNÇÕES MATEMÁTICAS E MÓDULO MATH

#### TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

- 1 Módulos
- 2 Funções *built-in* python
- 3 Importar módulos
- 4 Módulo math



# 1

## Módulos

- ❑ Um módulo consiste num bloco de código, geralmente constituído por um conjunto de funções que podemos importar e reutilizar no nosso código
- ❑ Diversos módulos estão integrados na *Python Standard Library*, fornecendo acesso a funcionalidades padrão da linguagem python
- ❑ A biblioteca padrão do Python (*Python Standard Library* ) faz parte de todas as instalações do Python. Inclui inúmeras funções (*built-in*), como:
  - ❑ *print*
  - ❑ *input*
  - ❑ *int*
  - ❑ *str*
  - ❑ *Float*
  - ❑ *...etc...*

# 1

## Módulos

- ❑ Funções *built-in* são funções internas e nativas, ou seja, que já vem incorporadas na linguagem e estão sempre disponíveis para utilização.
- ❑ Assim não é necessário a sua importação. Basta utilizá-las diretamente no código quando desejar.
- ❑ O Python possui inúmeras funções *built-in*, de diversas categorias.

## 2

## Funções *built-in*

<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

2

## Funções *built-in*

Algumas funções matemáticas, *Built-in Python*

Função	Descrição
max	Devolve o maior de um conjunto de literais
min	Devolve o menor de um conjunto de literais
abs	Devolve o valor absoluto
pow	Exponenciação, dada a base e o expoente
round	Devolve o valor arredondado

## 2 Funções *built-in*

```
1  # Algumas funções matemáticas incorporadas
2
3  val1 = 5
4  val2 = 10
5  val3 = 25
6
7  # Maior de um conjunto de literais
8  maior = max(val1, val2, val3)
9  print("\n maior =", maior)
10
11 # Menor de um conjunto de literais
12 menor = min(val1, val2, val3)
13 print("\n menor=", menor)
14
15 # valor absoluto
16 numero = -13.45
17 print("\n valor absoluto=", abs(numero))
18
19 numero1 = -13.45
20 numero2 = -12.55
21 # valor arredondado
22 print("\n arredondado às unidades", round(numero1, 0))
23 print("\n arredondado às decimas", round(numero1, 1))
24 print("\n arredondado às decimas", round(numero2, 1))
25
26 # expoente
27 numero = pow(3,2)
28 print("\n potencia de 3 ao quadrado", numero)
```

C:\WINDOWS\py.exe

maior = 25

menor= 5

valor absoluto= 13.45

arredondado às unidades -13.0

arredondado às decimas -13.4

arredondado às decimas -12.6

potencia de 3 ao quadrado 9

### 3 Importar módulos

❑ `import nomeModulo`

```
2 import math
3
```

```
import math      # módulo que incorpora funções matemáticas
import os        # módulo que incorpora funções d esistema operativo
import random    # módulo que incorpora funções de geração de números aleatórios
```



módulo a incorporar

- ❑ Quanto importamos mais do que módulo no mesmo programa, é boa prática importá-los por ordem alfabética



## 3 Importar módulos

❏ `import nomeModulo`

```
import random
num = random.randint(0,10)      # Return random integer in range [a, b], includes the end points
num1 = random.randrange(0,10)  # Return random integer in range [a, b[, excludes the end point
```

*módulo.função*

### 3 Importar módulos

❏ `from nomeMódulo import função`

```
from random import randint  
num = randint(0,10)      # Return random integer in range [a, b],
```



Importo apenas a função especificada, e não todo o modulo!

## 4

### Módulo *math*

- ❑ O módulo *math* incorpora um conjunto de funções matemáticas para estruturas de dados ditas simples
- ❑ Para estruturas de dados complexas, recorrer ao módulo *cmath*
- ❑ A generalidade das funções do módulo *math* devolve valores flutuantes (*float*)

4

## Módulo math

Algumas funções incorporadas no módulo math

Função	Descrição
ceil(x)	Arredonda para cima (menor inteiro acima de x)
floor(x)	Arredonda para baixo (maior inteiro abaixo de x)
trunc(x)	Trunca o valor x, devolve valor inteiro
fabs(x)	Valor absoluto de x
gcd(x,y)	Devolve o maior divisor comum entre x e y
sqrt(x)	Raiz quadrada de x
pi	Devolve a constante 3.14
sin, cos, tan	Funções trigonométricas

## 4 Módulo math

`ceil(x)` - Arredonda para cima (menor inteiro acima de x)

```
Exemplo_math.py > ...
1  import math      # importa o módulo math
2
3
4  numero = 13.25
5  numero1 = math.ceil(numero)
6  print("numero =", numero1)
7
8  numero = 13.55
9  numero1 = math.ceil(numero)
10 print("numero =", numero1)
11
12
```

C:\WINDOWS\py.exe

```
numero = 14
numero = 14
```

## 4 Módulo math

floor(x) - Arredonda para baixo (maior inteiro abaixo de x)

```
Exemplo_math.py > ...
1  import math      # importa o módulo math
2
3
4  numero = 13.25
5  numero1 = math.floor(numero)
6  print("numero =", numero1)
7
8  numero = 13.55
9  numero1 = math.floor(numero)
10 print("numero =", numero1)
11
12
```

C:\WINDOWS\py.exe

```
numero = 13
numero = 13
```

## 4 Módulo math

`trunc(x)` - Trunca o valor x, devolve valor inteiro

```
Exemplo_math.py > ...  
1  import math      # importa o módulo  math  
2  
3  
4  numero = 13.25  
5  numero1 = math.trunc(numero)  
6  print("numero =", numero1)  
7  
8  numero = 13.95  
9  numero1 = math.trunc(numero)  
10 print("numero =", numero1)  
11  
12
```

C:\WINDOWS\py.exe  
numero = 13  
numero = 13

## 4 Módulo math

`fabs(x)` – devolve o valor absoluto de x

```
Exemplo_math.py > ...  
1  import math      # importa o módulo  math  
2  
3  
4  numero = 13.25  
5  numero1 = math.fabs(numero)  
6  print("numero =", numero1)  
7  
8  numero = -13.95  
9  numero1 = math.fabs(numero)  
10 print("numero =", numero1)  
11  
12
```

C:\WINDOWS\py.exe  
numero = 13.25  
numero = 13.95



## 4 Módulo math

`sqrt(x)` – devolve a raiz quadrada de x

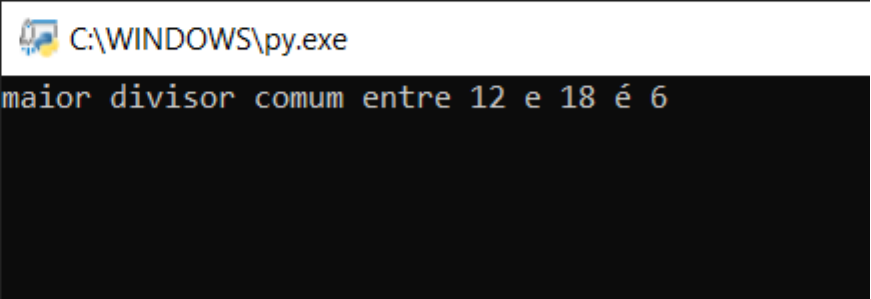
```
Exemplo_math.py > ...
1  # Algumas funções do módulo math
2  import math # importa o módulo math
3
4
5  numero = 144
6  raiz = math.sqrt(numero) # raiz quadrada de numero
7  print(raiz)
8
9
10
11
```

C:\WINDOWS\py.exe  
12.0

## 4 Módulo math

$\text{gcd}(x, y)$  – Devolve o maior divisor comum entre x e y

```
Exemplo_math.py > ...
1  # Algumas funções do módulo math
2  import math # importa o módulo math
3
4
5  numero1 = 12
6  numero2 = 18
7  divisor = math.gcd(numero1, numero2) # raiz quadrada de numero
8  print("maior divisor comum entre {0} e {1} é {2}" .format(numero1, numero2, divisor))
9
10
11
12
```



## 4 Módulo math

pi – Devolve o valor de pi

Exemplo\_math.py > ...

```
1  # Algumas funções do módulo math
2  import math # importa o módulo math
3
4
5  numero1 = 12
6  numero2 = 18
7  divisor = math.gcd(numero1, numero2) # raiz quadrada de numero
8  print("maior divisor comum entre {0} e {1} é {2}" .format(numero1, numero2, divisor))
9
10 print (math.pi)
11 print (round(math.pi, 2))
12
13
14
15
```

C:\WINDOWS\py.exe

maior divisor comum entre 12 e 18 é 6  
3.141592653589793  
3.14