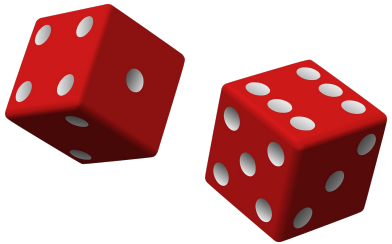
Conceitos Básicos

Dados-Informação-Conhecimento



- Dados
  - Elementos conhecidos de um problema.
  - Desprovidos de significado quando considerados isoladamente.

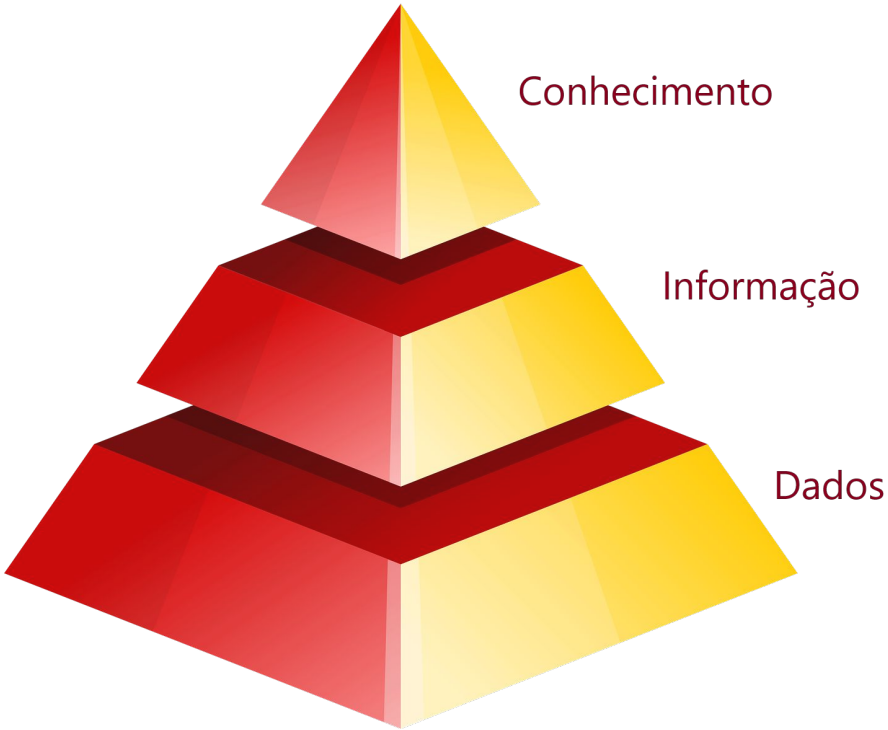
Dados-Informação-Conhecimento

- Informação
  - Um conjunto estruturado de dados.
  - Possuem utilidade e podem gerar ações.

Dados-Informação-Conhecimento

- Conhecimento
  - Síntese de múltiplas fontes de informação.
  - Possui elevado significado e utilidade para o suporte à tomada de decisões.

Dados-Informação-Conhecimento



Dados-Informação-Conhecimento

- Computador manipula informações contidas em sua memória.
- **Instruções**: comandam o funcionamento da máquina e determinam a maneira como os dados devem ser tratados.
- **Dados**: informação que devem ser manipulada pelo computador.

Tipos de Dados em Python

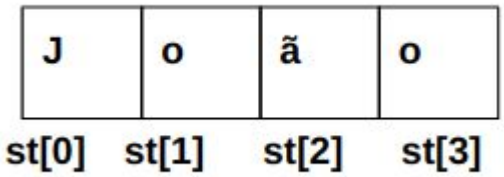
- Python utiliza o conceito de Objetos.
- Cada objeto pertence a um tipo.
  - Darth Vader é Sith.
  - Yoda é Jedi.
  - Chewbacca é Wookie.

Tipos de Dados em Python

- Programas manipulam *data objects*.
- Os objetos são divididos em:
  - Escalar: não podem ser subdivididos
    - numéricos: int e float.
    - lógico: bool (True e False)
    - NoneType: None.

Tipos de Dados em Python

- Não escalar: Possuem estrutura interna
    - Strings, tuplas e listas




```
t=(1,2.5,"Joao") a_l=[-1,2,3,[4,5]]
```

Manipulando Dados

- Comando type() retorna o tipo da variável:

```
>> type(34.9)
    float
>> type(2)
      int
```

Manipulando Dados

- Conversão de tipos

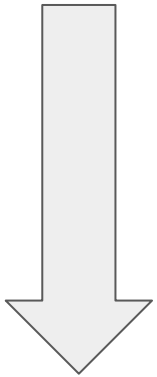
```
>> int (34.9)
    34
>> float(2)
      2.0
```

Manipulando Dados

- Operadores para int e float
  - Soma: a+b
    - Resulta int ou float
  - Produto: a*b
    - Resulta int ou float
  - Divisão: a/b
    - Resulta float
  - Divisão inteira: a//b
    - Resulta int (quociente inteiro)
  - Resto: a%b (módulo)
    - Resulta int (resto inteiro)
  - Produto: a**b (potência a^b)
    - Resulta int ou float

Manipulando Dados

- Expressões aritmética



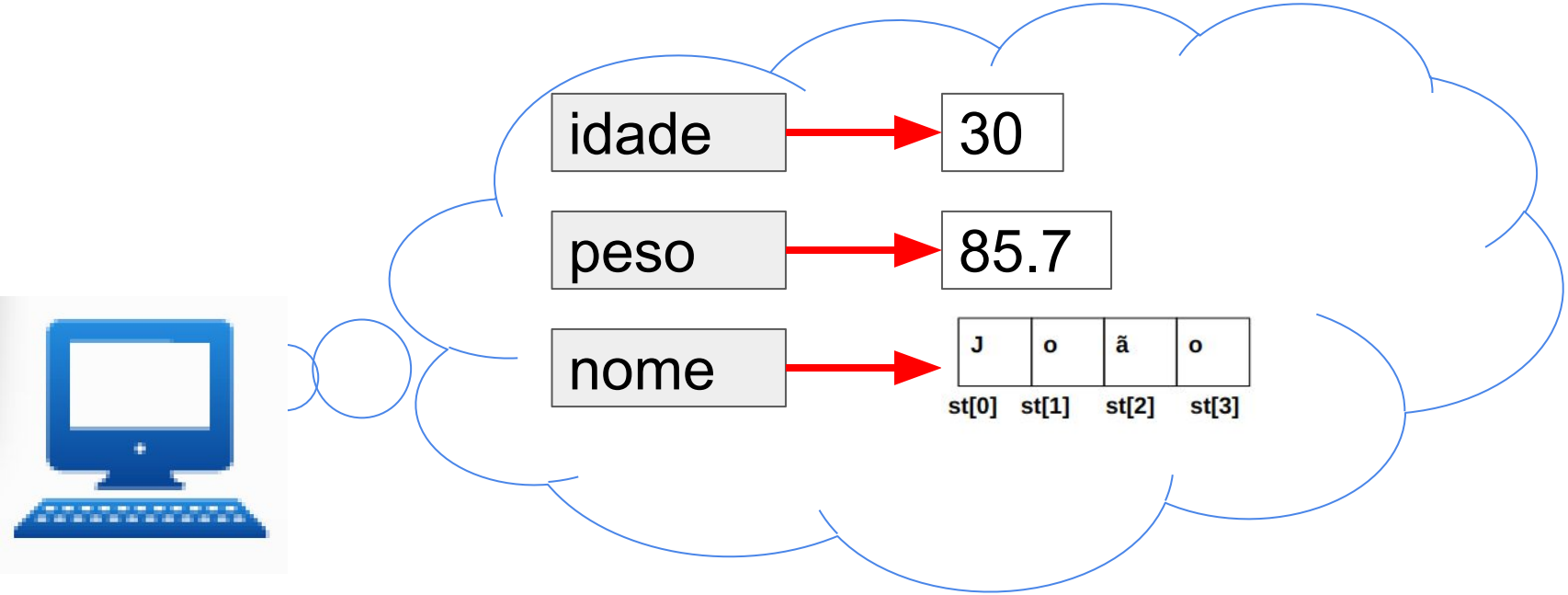
1. Parênteses
2. ** * / %
3. + -

- Operadores Relacionais

==	igual
!=	diferente
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

Variáveis

- As variáveis fornecem uma forma de associar nomes a objetos.



Variáveis

- Os dados estão sendo manipulados durante a execução do programa
- Os dados manipulados são armazenados na memória.
- As variáveis guardam informações sobre os dados (o seu conteúdo) que estão sendo manipulados.

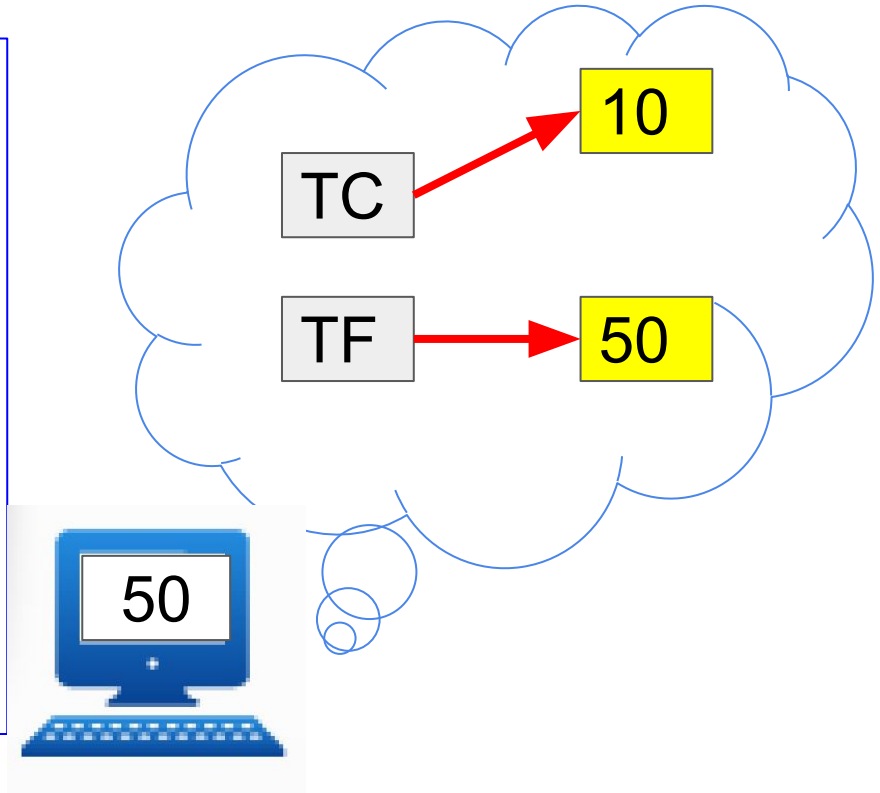
Variáveis

- O conteúdo da variável é substituído por outro que lhe será atribuído.
- O uso de uma variável em uma expressão representa o seu conteúdo naquele momento.
  - O uso não muda o seu conteúdo

Variáveis

```
TC = 10
TF = TC*9/5 + 32
print(TF)
```

```
TC = 35
TF=TC*9/5+32
print(TF)
```



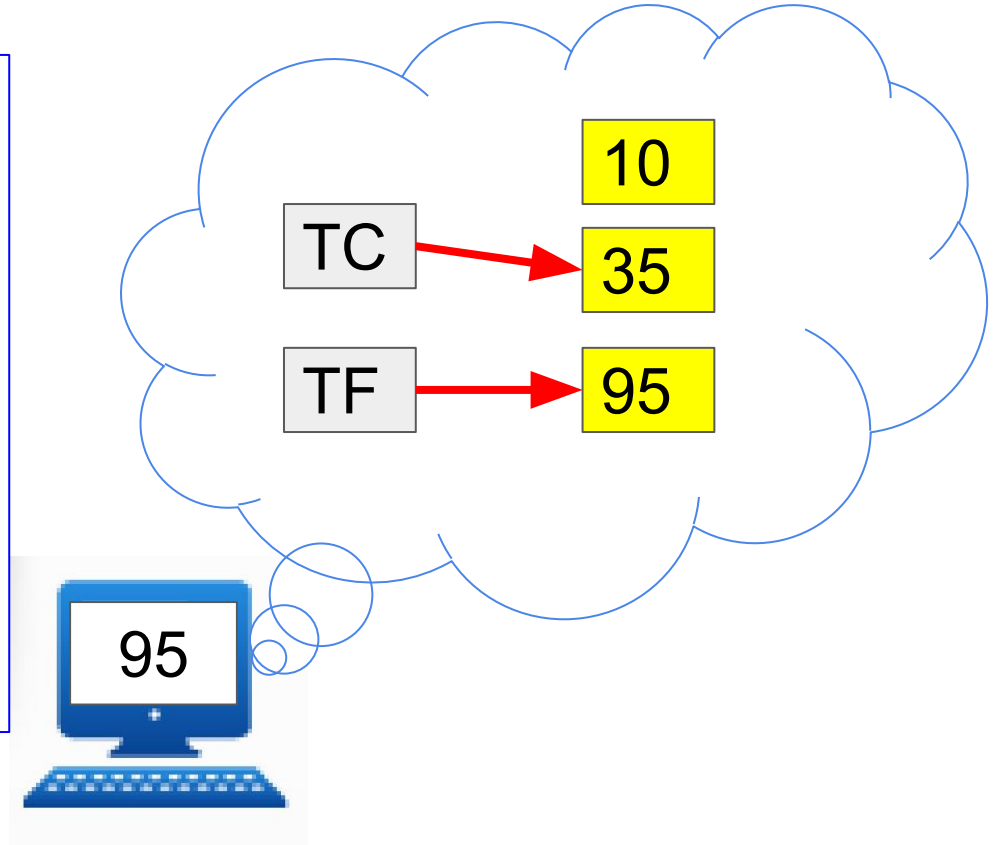
Variáveis

```
TC = 10
TF = TC*9/5 + 32
print(TF)
```

TC = 35

TF=TC*9/5+32

print(TF)



Variáveis

```
a,b = 10,20
print(a+b)          ⟹   30

a = a+1
print(a)            ⟹   11
a += 1
print(a)            ⟹   12

print('a= ', a, 'b=', b)   ⟹   a= 12 b= 20
```

Variáveis

- Nomes de variáveis podem conter letras maiúsculas e minúsculas, dígitos (mas não podem começar com um dígito) e o caractere especial _

nome_pessoa = ‘João’ 

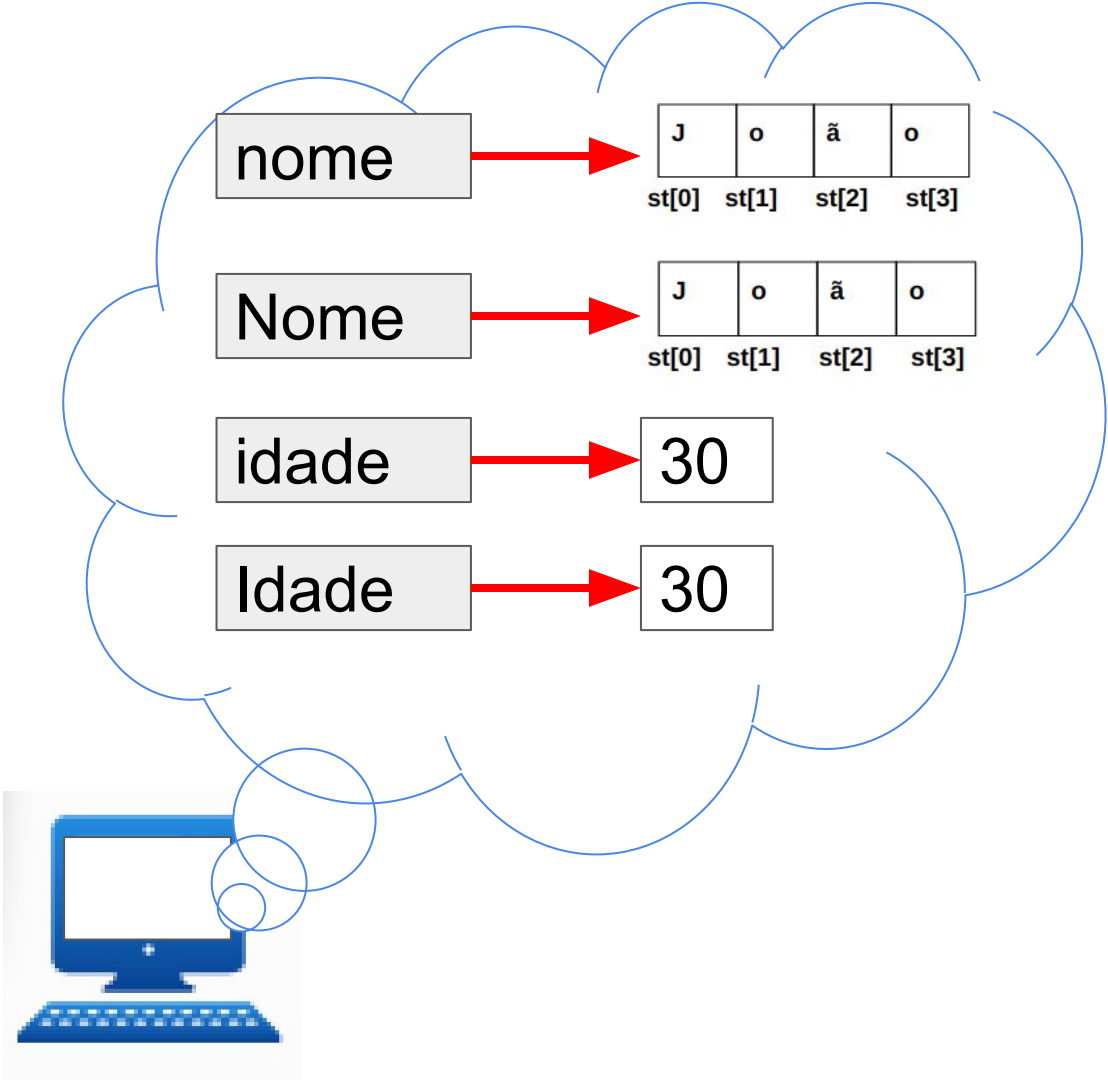
idadePessoa_01 = 30 

01idadePessoa = 30 

Variáveis

- Python é case-sensitive

nome = ‘João’
Nome = ‘João’
idade= 30
Idade = 30



Variáveis

- O nome de uma variável facilita o entendimento do código.

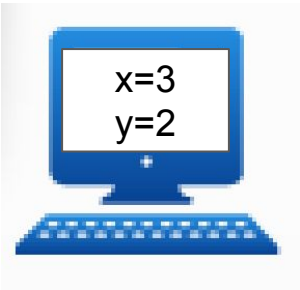
```
a = ‘João’
b = 30
c = 85.7
```

```
nome = ‘João’
idade = 30
peso = 85.7
```

Variáveis

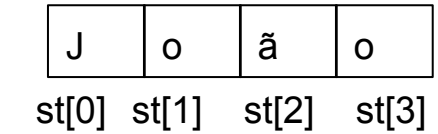
- Python permite atribuições múltiplas

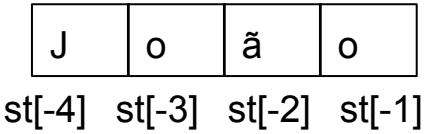
```
a,b = ‘João’,30

x, y = 2, 3
x, y = y, x
```



Strings - Cadeia de caracteres

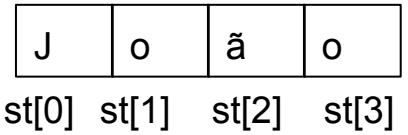
- Cadeia de caracteres que diferenciam maiúsculos de minúsculos
  - “Ana” ≠ “ana”
  - “ana” > “Ana”
  - “João” > “Ana”
- Declara-se string com “ ” ou ‘’.
  - nome = “joao” nome = ‘joao’
- Acesso através de índices:
  - st=”João”

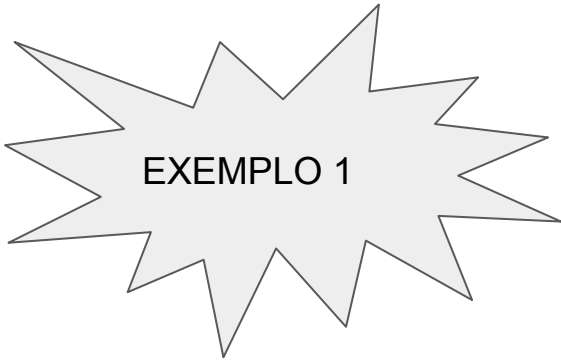




Strings - Cadeia de caracteres

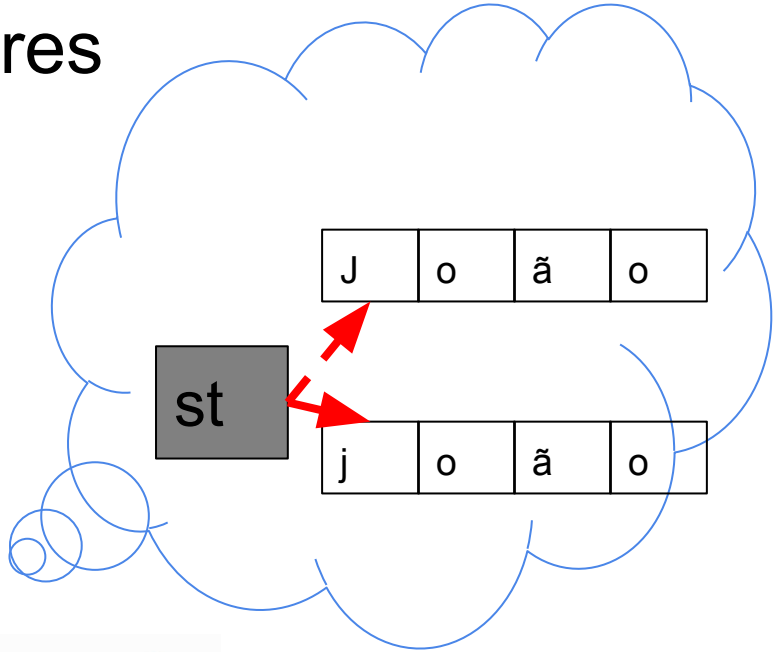
- Tamanho da cadeia de caracteres
  - len(st) ⟹ 4
- Percorrendo a cadeia de caracteres através dos índices
  - st[1:4] ⟹”oão””
  - st[1:4:2] ⟹”oo””
  - st[: :] ⟹”João””
  - st[0:len(str):1] ⟹”João””
  - st[: : -1] ⟹”oãoJ””
  - st[-1:- **(len(str)+1)** : -1] ⟹”oãoJ””
  - st[3:1:-1] ⟹”ãoJ””





Strings - Cadeia de caracteres

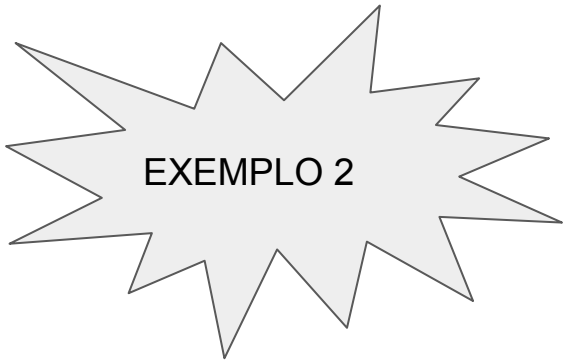
J	o	ã	o
st[0]	st[1]	st[2]	st[3]



- Strings podem ser concatenadas (+) mas não podem ser pontualmente modificadas
  - st[0]=’j’ ⇒ERRO!!
  - st = ‘j’+st[1:len(st)] ⇒”joão” Ok!!

String - Operadores

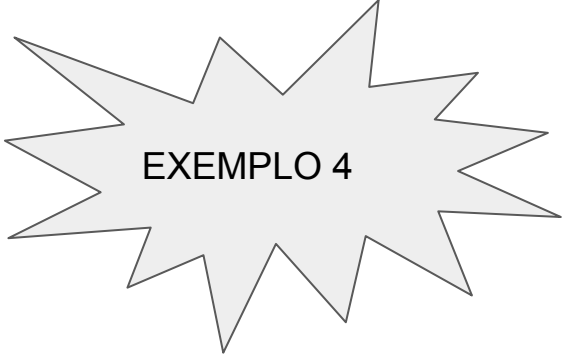
```
s1="olá"
s2="mundo"
s3=s1+s2
print(s3)                ⇒ “olámundo”
s3=s1+" "+s2+"!"
print(s3)                ⇒ “olá mundo!”
print(s1*3)              ⇒ “oláoláolá”
print(s1 in s3)          ⇒ True
print(s2 in s3)          ⇒ True
print('oi' in s3)        ⇒ False
print('oi' not in s3)    ⇒ True
```



String - Funções nativas

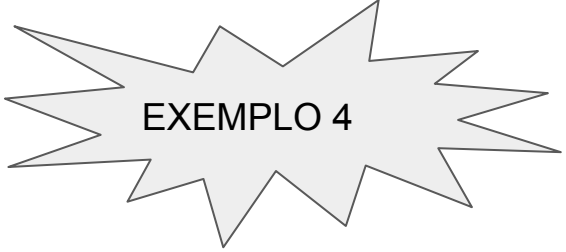
- **ord(c):** retorna o valor inteiro correspondente ao caracter **c** na tabela ASCII (American Standard Code for Information Interchange).
- **chr(n):** retorna o caracter correspondente ao inteiro **c** na tabela ASCII. Inverso de ord(c).
- **len(st):** retorna o tamanho da cadeia de caracteres.
- **str(obj):** retorna a representação em cadeia de caracteres do objeto **obj**.

String - Alguns métodos



- s = ‘uma string’
- s.lower(): retorna a string com todos os caracteres minúsculos.
- s.upper(): retorna a string com todos os caracteres maiúsculos.
- s.swapcase(): retorna a string invertendo os caracteres de maiúsculo para minúsculo e vice-versa.
- s.title(): retorna a string no formato de título.
- s.capitalize(): retorna a string com o primeiro caractere maiúsculo e os demais minúsculos.

String - Alguns métodos



- s.capitalize(): retorna a string com o primeiro caractere maiúsculo e os demais minúsculos.
- s.count(<subString>, <início>, <fim>): retorna quantas vezes a substring aparece entre o valor início e fim na cadeia de caracteres.
- s.find(<subString>, <início>, <fim>): retorna o menor índice onde a substring ocorre ou -1 se não for encontrada.
- …..várias outras

Entrada de Dados

- input(“”): retorna uma string fornecida via teclado

```
nome= input(“digite um nome:”)
print(nome)
```

- Precisa converter para outro tipo.

```
idade = int(input(“digite idade:”))
print(idade)
altura = float(input(“digite altura:”))
print(altura)
```

Entrada de Dados

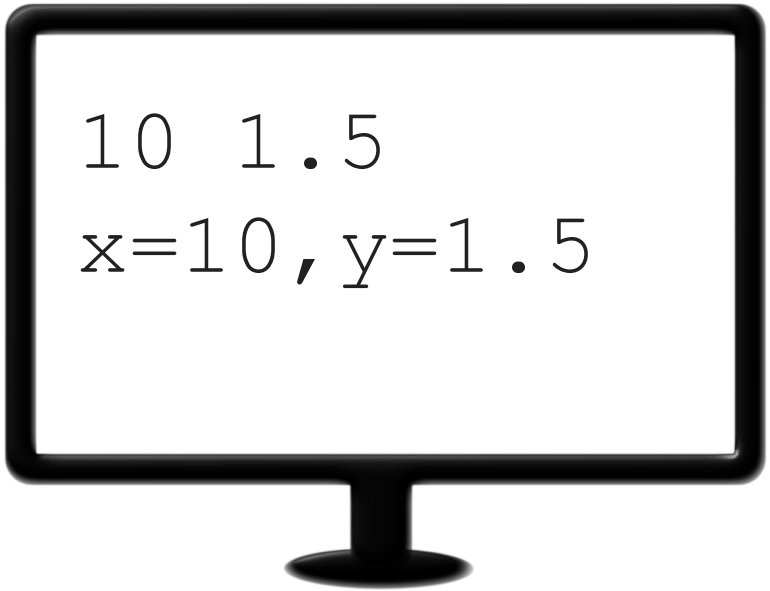
- O programa fica aguardando o usuário fornecer uma entrada.
- input() converte o que for digitado em uma string.
- O usuário deve realizar o typecasting, se quiser converter de string para outro tipo.
- A mensagem exibida usando input() é opcional.

```
idade = int(input(“digite idade:”))
print(idade)
altura = float(input(“digite altura:”))
print(altura)
```

Entrada de Dados

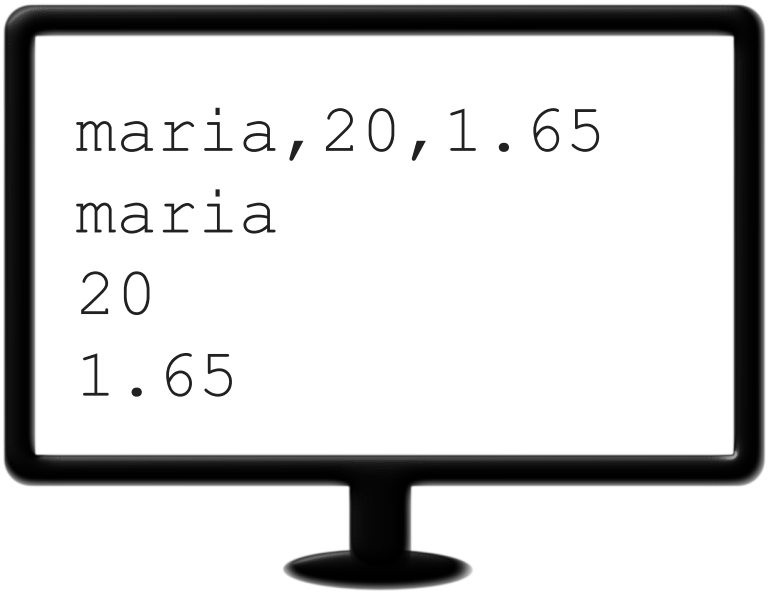
- Múltiplas entradas podem ser processadas via métodos map() e split() .
- split() separa as entradas separadas por espaço (default), retornando um objeto iterável.

```
x, y = input().split()
x=int(x)
y=float(y)
print(f'{x=},{y=}')
```



Entrada de Dados

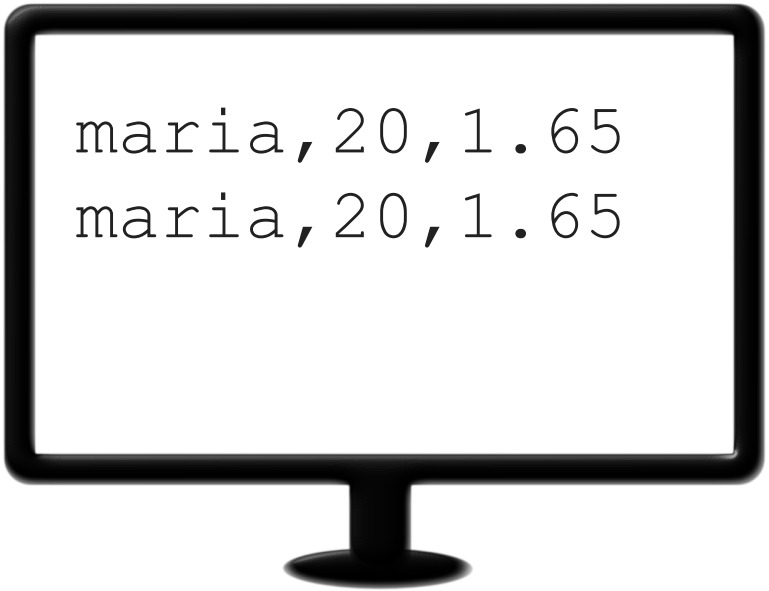
- Múltiplas entradas podem ser processadas via métodos map() e split() .
- split() separa as entradas separadas por espaço (default), retornando um objeto iterável.




```
nome,idade,altura = input().split()
idade = int(idade)
altura = float(altura)
print(f'{nome}\n{idade}\n{altura}')
```

Entrada de Dados

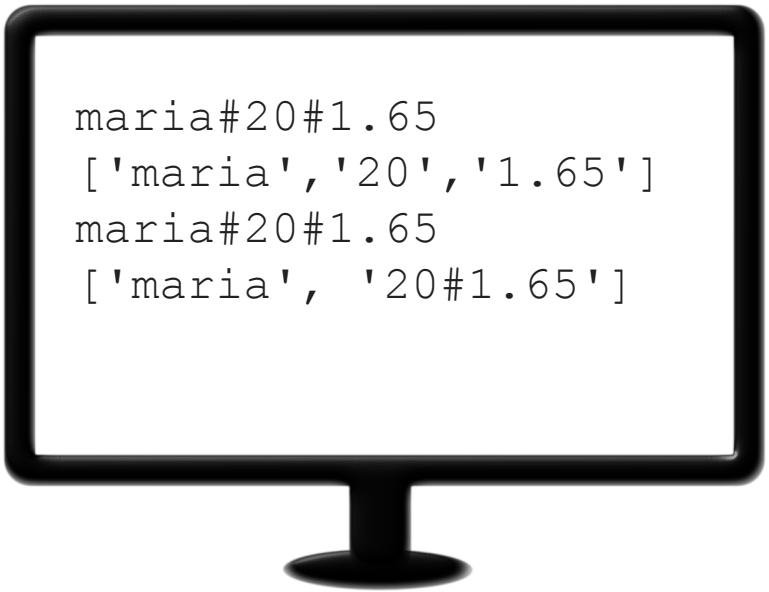
- Múltiplas entradas podem ser processadas via métodos map() e split() .
- split() separa as entradas separadas por espaço (default), retornando um objeto iterável.




```
nome,idade,altura = input().split(“,”)
idade = int(idade)
altura = float(altura)
print(f'{nome}\n{idade}\n{altura}')
```

Entrada de Dados

- Múltiplas entradas podem ser processadas via métodos map() e split() .
- split() separa as entradas separadas por espaço (default), retornando um objeto iterável.

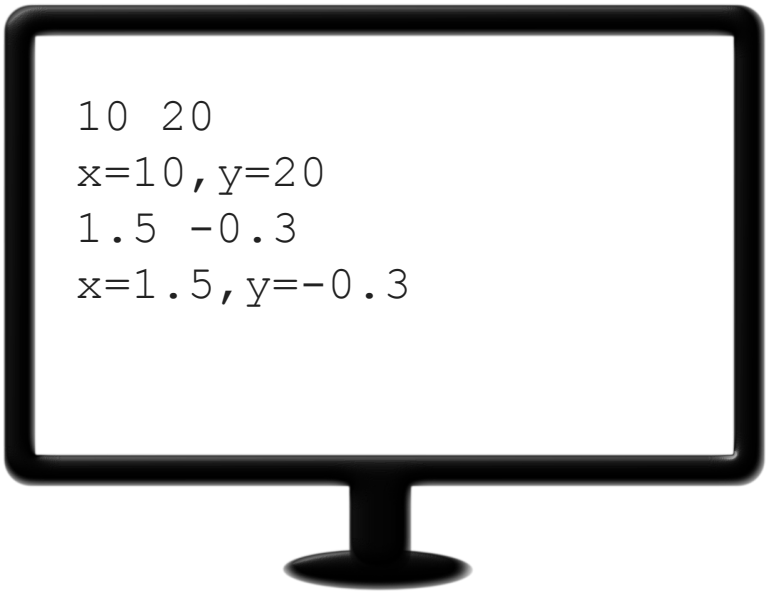



```
dados = input().split(“#”)
print(dados)
dados = input().split(“#”,1)
print(dados)
```

Entrada de Dados

- map(<função>,<iterável>): retorna um objeto com os resultados obtidos após aplicar <função> a cada item de um determinado <iterável>
    - <função>: função que recebe e processa cada elemento em <iterável>.
    - <iterável>: objeto a ser mapeado que pode ser uma lista, tupla, etc.

Entrada de Dados




```
x, y = map(int, input().split())
print(f'{x=},{y=}')
x, y = map(float, input().split())
print(f'{x=},{y=}')
```

String - Formatted String Literal f-string

- Uma breve ideia sobre f-string

  x = int(input('x=')) ⟹ x=10

  y= int(input('y=')) ⟹ y=20

  print(x,'+',y,'=',x+y) ⟹ 10 + 20 = 30

  print(f'**{x}**+**{y}**=**{x+y}**') ⟹ 10+20=30