# Comandos de Entrada e Saída

Prof. Alberto Costa Neto

Programação em Python

# Entrada de Dados

- Sugerimos o uso da função input
  - Ela recebe uma String como parâmetro. Este parâmetro será impresso na tela e o cursor ficará aguardando que algo seja digitado via teclado.

```
valor = input('Digite seu Nome:')
```

  - A linha acima irá imprimir na tela

```
Digite seu Nome:_
```

  - E o cursor ficará aguardando que algo seja digitado após o caracteres dois pontos

# Entrada do Usuário

Ao executar a função input(),
o cursor ficará piscando
esperando que algo seja
digitado e se aperte a teclar
<ENTER>

Apenas após isso é que o
valor digitado será
armazenado na variável valor
(do tipo String)

```
nome = input('Quem é você?')
print('Bem-vindo', nome)
```

Quem é você? Alberto
Bem-vindo Alberto

# Entrada de Strings

- A função input pega tudo que foi digitado na linha e retorna uma String, mesmo que tenha espaços ou números no meio

```
valor = input()
```

- Exemplos de entrada e valores atribuídos à variável valor:
  Este foi o valor digitado
  Digitei 25.0 e 10
  12.55

valor = 'Este foi o valor digitado'

valor = 'Digitei 25.0 e 10'

Valor = '12.55'

# Entrada de Números Inteiros (int)

- A função input() retorna uma String. Como posso ler um inteiro? Simples! Transforma-se o conteúdo da String em um número inteiro (em Python, são representados pelo tipo int)

```
valor = input()
valor_inteiro = int(valor)
```

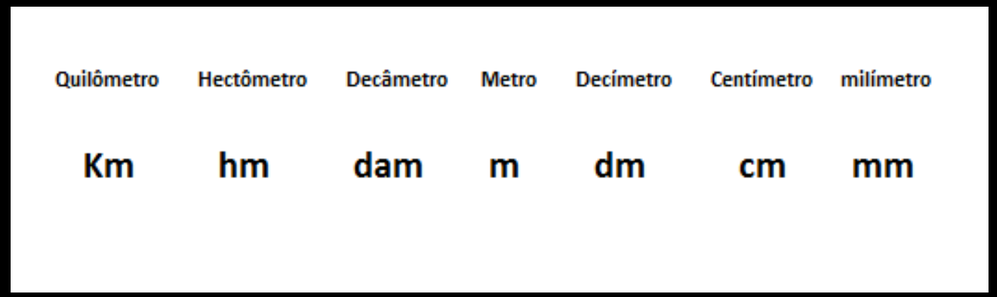
- valor continua sendo um valor String. Mas a variável valor_inteiro contém o inteiro que corresponde aos caracteres numéricos contidos na String. De forma resumida, poderíamos escrever apenas:

```
valor_inteiro = int(input())
```

# Convertendo a Entrada do Usuário

```
d_km = input('Distância em Km:')
d_m = int(d_km) * 1000
print('Distância em Metros', d_m)
```

Exemplo:
Convertendo de km para metros




Distância em Km: 15
Distância em Metros: 15000

# Entrada de Números em Ponto Flutuante (float)

- De forma semelhante aos inteiros, os números em ponto flutuante são lidos como String usando a função input() e em seguida são convertidos para o tipo float (ponto flutuante/real).

```
valor = input()
valor_float = float(valor)
```

ou

```
valor_float = float(input())
```

# Vários dados em uma mesma linha da entrada

- Uma questão pode pedir, por exemplo, para ler a matrícula e 2 notas para calcular sua média final.
- Como fazer a separação dos dados se a função input lê uma linha e retorna uma única String?

```
>>> entrada = input()
2015021234 10.0 9.0
>>> print(entrada)
2015021234 10.0 9.0
>>> valores = entrada.split()
>>> print(valores)
['2015021234', '10.0', '9.0']
>>> matricula = int(valores[0])
>>> print(matricula)
2015021234
>>> nota1 = float(valores[1])
>>> print(nota1)
10.0
>>> nota2 = float(valores[2])
>>> print(nota2)
9.0
```

# Saída de Dados

- Você pode fazer o programa todo correto, mas se não imprimir a saída conforme pedido no problema, o sistema considera a questão errada.
- Portanto, é importantíssimo estar atento a como a saída deve ser apresentada.
- Para executar a impressão de dados na tela, sugerimos usar a função print
  - Ela recebe um ou mais valores separados por vírgula e imprime na tela os valores separados por espaços em branco.

# Imprimindo ponto flutuante com casas decimais

- Muitas questões pedem para imprimir um número real (ponto flutuante) com um certo número de casas decimais.
- Neste caso sugerimos usar o operador % que no caso de Strings, serve para formatação (no estilo C)

```
>>> #Importando biblioteca math
... import math
>>> print(math.pi)
3.14159265359
>>> #6 caracteres com 4 casas decimais
... print('%6.4f' % math.pi)
3.1416
>>> #6 caracteres com 3 casas decimais
... print('%6.3f' % math.pi)
 3.142
>>> print('%7.3f' % math.pi)
  3.142
>>> #3 casas decimais
... print('%.3f' % math.pi)
3.142
```

# Imprimindo vários dados em uma mesma linha da saída

- Sugerimos o uso da função print
  - Ela recebe um ou mais valores separados por vírgula e imprime na tela os valores separados por espaços em branco.
  - Uma alternativa é concatenar Strings e passar um único parâmetro

```
>>> print('Ok')
Ok
>>> print('Valor =', 'Ok')
Valor = Ok
>>> print('Valor = ' + 'Ok')
Valor = Ok
>>> print('Valor =', 10)
Valor = 10
>>> print('Valor = ' + 10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> print('Valor = ' + str(10))
Valor = 10
```