# Aplicações de Listas

Prof. Alberto Costa Neto

Programação em Python

# Usos comuns de laços
# E de listas também!

1. Encontrar o maior/menor valor entre vários
2. Somar um conjunto de valores
3. Calcular a média de um conjunto de valores
4. Buscar por um valor entre outros valores
5. Filtrar um conjunto de valores
6. Ordenar valores

# maior, menor, soma e contagem com funções built-in para Listas

- Existem várias funções predefinidas (built-in) em Python que aceitam listas como parâmetro
- Lembram dos usos dos laços? Estas funções são muito mais simples!
- Permitem obter maior, menor, soma e contar

```
>>> nums = [3, 41, 12, 9, 74, 14]
>>> print(len(nums)) # contar
6
>>> print(max(nums)) # maior
74
>>> print(min(nums)) # menor
3
>>> print(sum(nums)) # somar
153
>>> print(sum(nums)/len(nums)) # média
25.5
```

# Calculando a média dos valores lidos usando uma lista

Usa variáveis para contar e acumular para no final calcular a média

```
total = 0
cont = 0
while True :
    valor = input('Numero: ')
    if valor == 'fim' : break
    num = float(valor)
    total = total + num
    cont = cont + 1

media = total / cont
print('Media:', media)
```

```
Numero: 3
Numero: 9
Numero: 5
Numero: fim
Média: 5.66666666667
```

```
numlist = list()
while True :
    valor = input('Numero: ')
    if valor == 'fim' : break
    num = float(valor)
    numlist.append(num)

media = sum(numlist) / len(numlist)
print('Media:', media)
```

Armazena os valores em uma lista para calcular a média no final usando funções

# Filtrar valores: Encontrar valores maiores que a média usando Lista

```
numlist = list()
while True :
    valor = input('Numero: ')
    if valor == 'fim' : break
    num = float(valor)
    numlist.append(num)

media = sum(numlist) / len(numlist)
print('Media:', media)
for n in numlist:
   if n > media:
      print('Maior que a média:', n )
```

```
Numero: 3
Numero: 9
Numero: 5
Numero: 4
Numero: 6
Numero: fim
Média: 5.4
Maior que a média: 9.0
Maior que a média: 6.0
```

# Melhores Amigos: Strings e Listas

```
>>> abc = 'Com tres palavras'
>>> palavras = abc.split()
>>> print(palavras)
['Com', 'tres', 'palavras']
>>> print(len(palavras))
3
>>> print(palavras[0])
Com
```

```
>>> print(palavras)
['Com', 'tres', 'palavras']
>>> for p in palavras :
...     print(p)
...
Com
tres
palavras
>>>
```

split quebra uma string em partes e produz uma lista de strings. Podemos pensar em algo análogo a quebrar uma frase em uma lista de palavras. Podemos acessar uma palavra específica ou iterar sobre todas as palavras

```
>>> linha = 'Um monte                    de espacos'
>>> palavras = linha.split()
>>> print(palavras)
['Um', 'monte', 'de', 'espacos']
>>>
>>> linha = 'primeiro;segundo;terceiro'
>>> algo = linha.split()
>>> print(algo)
['primeiro;segundo;terceiro']
>>> print(len(algo))
1
>>> algo = linha.split(';')
>>> print(algo)
['primeiro', 'segundo', 'terceiro']
>>> print(len(algo))
3
>>>
```

- Quando não é especificado um delimitador, múltiplos espaços são tratados como um delimitador
- Podemos especificar qual caractere delimitador desejamos usar passando-o como parâmetro na chamada a split