

# Pensamento Computacional e Lógica de Programação

Prof. Dr. Álvaro Campos  
Ferreira

# Web Scraping

Dados obtidos da página estão na linguagem HTML, que divide os dados no que chama de tags.

Para explorar as tags de um site, usa-se a ferramenta de desenvolvimento do navegador, F12 no Firefox.

# Web Scraping

Uma vez identificadas as tags de interesse, filtramos o resultado:

```
soup.findAll('div')  
for tag in soup.findAll('div'):  
    print(tag.text)
```

# Qualidade de dados

Dimensões da qualidade de dados:

- Completude ou integridade
- Conformidade
- Validade
- Acurácia e precisão

# Valores faltantes

Valores faltantes são representados por **nan** em DataFrames e não são a mesma coisa que valores nulos.

- Faltas (completamente) aleatórias
- Faltas não aleatórias

# Tipos das variáveis

# Tipos de variáveis

Para cada tipo de variável, as funções tem comportamento diferente.

- Texto (str)
- Número inteiro (int)
- Número real (float)
- Booleano (bool), True ou False

# Tipos de variáveis

Ao declarar uma variável, o Python determina seu tipo dinamicamente, uma variável pode mudar de tipo apenas declarando-a novamente:

```
idade = "Trinta e dois" # tipo texto
```

```
idade = 32 # tipo inteiro
```



# Listas

# Declarando listas

Uma lista pode ter vários argumentos de diferentes tipos.

```
minhaLista = ['eu', 'amo', 'Python']  
minhaLista[0]
```

# Declarando listas

Modificando elementos de uma lista

```
minhaLista = ['eu', 'amo', 'Python']  
minhaLista[2] = 'software livre'
```

# Dicionários

# Dicionários

Declarando um dicionário com duas entradas e acessando cada uma delas:

```
meuDict = {'linguagem': 'Python',  
'idade': '28 anos'}
```

```
meuDict['linguagem']
```

```
meuDict['idade']
```

# Funções e bibliotecas

# Funções

Algumas funções importantes para Listas:

- append()
- insert()
- del
- pop()
- remove()
- count()
- reverse()
- sort()
- index()

# Funções

Funções importantes para Dicionários:

- clear()
- copy()
- items()
- keys()
- pop()
- update()
- values()



# Controle de fluxo

# Laços

- Em português, chamamos “loop” de “laço”
- Palavra-chave **for**

```
for cliente in clientes:
```

```
    print(cliente)
```

```
    if cliente == “Álvaro”:
```

```
        print(“Te peguei!”)
```

# Laços for ou “para cada”

Com **for**, o bloco irá executar para cada elemento do objeto.

```
for cliente in clientes:  
    print(cliente)
```

# Laços while ou “enquanto”

O **while** executará o bloco “enquanto” a condição a direita dele resultar em verdadeiro, ou booleano True.

```
while nome == “Álvaro”:  
    print(“Olá, professor!”)
```

# Blocos if... else ou “se... senão”

O **if** ou “se”, executa o bloco se a condição for satisfeita.

```
if nome == “Álvaro”:
```

```
    print(“Bom dia, professor”)
```

```
else:
```

```
    print(“Bom dia, querido colega”)
```

# Funções

# Funções

Define-se funções com a palavra-chave **def** da palavra definir:

```
def somar_numeros(a,b):  
    return a + b
```

# Retornando valores

Uma função pode retornar um ou mais valores após a execução.

```
def somar_numeros(a,b):  
    """ Soma dois números.  
    """  
    return a + b
```



# Módulos

# Módulos de funções

```
def main():  
    print("Meu módulo é o máximo.")  
if __name__ == '__main__':  
    main()
```

# Encapsulando funcionalidades

# Importando bibliotecas

# Importando bibliotecas

A função `random()` da biblioteca `random` gera um número aleatório entre 0 e 1.

```
import random  
random.random() # será entre 0 e 1
```

# Importando bibliotecas

A função `randint()` da biblioteca `random` gera um número inteiro aleatório.

```
import random  
random.randint(0,2) # será 0, 1 ou 2
```



INSTITUTO BRASILEIRO DE ENSINO,  
DESENVOLVIMENTO E PESQUISA