

# FUNDAMENTOS DE BIG DATA E DATA ANALYTICS COM PYTHON

**Professor:** Eduardo Inocencio

# INTRODUÇÃO AO PANDAS

Pandas é uma biblioteca do Python muito utilizada em programação científica. Pandas trouxe um *plus* ao Python, pois possibilita trabalhar com análise de dados sem ter que recorrer a outras linguagens.



Pandas = **PAN**el **DA**ta**S**

# POR QUE USAR PANDAS?

Pandas é uma biblioteca Python usada para trabalhar com conjuntos de dados.

Possui funções para analisar, limpar, explorar e manipular dados.

# O QUE PODEMOS FAZER COM PANDAS?

- **Manipulação de dados:** de forma rápida, ágil e com indexação integrada.
- **Análise de dados:** leitura, escrita, alinhamento, reshaping, slicing, agrupamentos, fusão, concatenação...
- **Variedade de uso:** mercado financeiro, neurociência, economia, estatística, publicidade, e muito mais...

# PRIMEIROS PASSOS

Professor: Eduardo Inocencio

# INSTALAÇÃO DE PANDAS

Se você já possui [Python](#) e [PIP](#) instalados em um sistema, a instalação do Pandas é muito fácil.

Instale-o usando este comando:

```
C:\Users\Your Name>pip install pandas
```

# IMPORTANDO PANDAS

Importando a biblioteca.

```
import pandas as pd
```

Pandas não é uma biblioteca *built-in*, então é preciso instalá-la, caso a instalação do Python tenha sido feita sem o Anaconda.

# EXEMPLO

```
import pandas

mydataset = {
    'cars': ["BMW", "Volvo", "Ford"],
    'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)

print(myvar)
```



# PANDAS COMO PD

Pandas geralmente é importado com o **pd** pseudônimo.

**alias:** Em Python, alias é um nome alternativo para se referir à mesma coisa.

Crie um alias com a **as** palavra-chave durante a importação:

```
import pandas as pd
```

# EXEMPLO

```
import pandas as pd

mydataset = {
    'cars': ["BMW", "Volvo", "Ford"],
    'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

# VERIFICANDO A VERSÃO DO PANDAS

A string da versão é armazenada no `__version__` atributo.

```
import pandas as pd

print(pd.__version__)
```

# PANDAS SERIES

Professor: Eduardo Inocencio

# O QUE É UMA SÉRIE?

Uma série em Pandas é como uma coluna de uma tabela.

É uma matriz unidimensional que contém dados de qualquer tipo. Podemos usar a função `pd.Series()` para criar uma série

```
idades = pd.Series([15,20,30,40,50,60])  
print(idades)
```

Como parâmetro, colocamos uma lista de valores.

# EXEMPLO

Crie uma série Pandas simples a partir de uma lista:

```
import pandas as pd  
  
a = [1, 7, 2]  
  
myvar = pd.Series(a)  
  
print(myvar)
```

# ÍNDICES

Se nada mais for especificado, os valores serão rotulados com seu número de índice. O primeiro valor tem índice 0, o segundo valor tem índice 1 etc.

Este rótulo pode ser usado para acessar um valor especificado.

## Exemplo

Retorne o primeiro valor da Série:

```
print(myvar[0])
```

# CRIANDO ÍNDICES

Com o **index** argumento, você pode nomear seus próprios rótulos.

## Exemplo

Crie seus próprios rótulos:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```



# EXEMPLO

```
index = ["João", "Ana", "Maria", "Jorge", "Rose", "Jorge"]  
idades.index = index  
  
print(idades)
```

```
João      15  
Ana       20  
Maria     30  
Jorge     40  
Rose      50  
Jorge     60  
dtype: int64
```

# EXEMPLO

PARA CONFIRMAR OS ÍNDICES:

```
idades.index
```

```
Index(['João', 'Ana', 'Maria', 'Jorge', 'Rose', 'Jorge'], dtype='object')
```

PARA OBTER APENAS OS VALORES:

```
idades.values
```

```
array([15, 20, 30, 40, 50, 60], dtype=int64)
```

# ACESSANDO INFORMAÇÕES

Depois de criar etiquetas, você pode acessar um item consultando a etiqueta.

## Exemplo

Retorne o primeiro valor da Série:

```
print(myvar["y"])
```

# OBJETOS CHAVE/VALOR COMO SÉRIE

Você também pode usar um objeto chave/valor, como um dicionário, ao criar uma série.

## **Exemplo:**

Crie uma série Pandas simples a partir de um dicionário:

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

**Nota:** As chaves do dicionário tornam-se os rótulos.

# EXEMPLO

Para selecionar apenas alguns itens do dicionário, use o `index` argumento e especifique apenas os itens que deseja incluir na Série.

## Exemplo

Crie uma série usando apenas dados de "dia1" e "dia2":

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

# PANDAS DATAFRAME

Professor: Eduardo Inocencio

# O QUE É UM DATAFRAME?

Um Pandas **DataFrame** é uma estrutura de dados bidimensional, como um array bidimensional ou uma tabela com linhas e colunas.

## Exemplo

Crie um DataFrame simples do Pandas:

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

#load data into a DataFrame object:

df = pd.DataFrame(data)

print(df)
```

# O QUE É UM DATAFRAME?

Podemos criar *dataframes* a partir da função `pd.DataFrame()`, utilizando como argumento tanto um *array*, quanto usando dicionários.

```
df = pd.DataFrame(np.array([[1,2,3],[4,5,6],[7,8,9]]), columns = ['a','b','c'])  
df
```

	a	b	c
0	1	2	3
1	4	5	6
2	7	8	9



# CRIANDO DATAFRAMES

## USANDO DICIONÁRIOS.

```
dados = {'valor1':[1,2], 'valor2':[3,4], 'valor3':[5,6]}  
  
df2 = pd.DataFrame(data=dados)  
df2
```

	valor1	valor2	valor3
0	1	3	5
1	2	4	6

## USANDO DICIONÁRIOS E EXPLICITANDO OS ÍNDICES.

```
dados = {'valor1':[1,2], 'valor2':[3,4], 'valor3':[5,6]}  
  
df3 = pd.DataFrame(data=dados, index = ['a','b'])  
df3
```

	valor1	valor2	valor3
a	1	3	5
b	2	4	6

# DATAFRAMES – ACESSANDO DADOS

Similar ao caso da Series, podemos acessar tanto os valores quanto os índices do *Dataframe* com os métodos *pd.values* e *pd.index*.

```
df3.values
```

```
array([[1, 3, 5],  
       [2, 4, 6]], dtype=int64)
```

Ainda podemos obter os nomes das colunas com o *pd.columns*

```
df3.columns
```

```
Index(['valor1', 'valor2', 'valor3'], dtype='object')
```

# LOCALIZAR LINHA

Como você pode ver no resultado acima, o DataFrame é como uma tabela com linhas e colunas.

Pandas usam o **loc** atributo para retornar uma ou mais linhas especificadas

# EXEMPLO

Linha de retorno **0**:

```
#refer to the row index:  
print(df.loc[0])
```

Retornar linha **0** e **1**:

```
#use a list of indexes:  
print(df.loc[[0, 1]])
```

**Nota:** Ao usar **[]**, o resultado é um Pandas **DataFrame** .

# ÍNDICES NOMEADOS

Com o argumento **index**, você pode nomear seus próprios índices.

## Exemplo

Adicione uma lista de nomes para dar um nome a cada linha:

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

# LOCALIZE ÍNDICES NOMEADOS

Use o índice nomeado no atributo **loc** para retornar as linhas especificadas.

## Exemplo

Retorne "dia2":

```
#refer to the named index:  
print(df.loc["day2"])
```

# CARREGAR ARQUIVOS EM UM DATAFRAME

Se seus conjuntos de dados estiverem armazenados em um arquivo, o Pandas poderá carregá-los em um **DataFrame**.

## Exemplo

Carregue um arquivo separado por vírgula (arquivo CSV) em um DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

# CARREGAR ARQUIVOS EM UM DATAFRAME

A biblioteca Pandas é capaz de ler diversos tipos de arquivos, com uma sintaxe simples. Dentre os tipos de arquivos que podemos ler com o Pandas, temos:

- `read_table`
- `read_csv`
- `read_excel`
- `read_hdf`
- `read_sql`
- `read_json`
- `read_html`
- `read_stata`
- `read_sas`



# EXEMPLOS DE LEITURA

Pandas é capaz de ler arquivos, com uma sintaxe muito simples. Veja alguns exemplos:

```
df = pd.read_excel('bank-full.xlsx', sheet_name='Dados')  
df = pd.read_table('bank-full.txt', delim_whitespace=True)  
df = pd.read_csv('bank-full.csv')
```

Há diversos parâmetros que podem ser adicionados ao comando de leitura.

Veja a documentação de cada um sempre que precisar!

# MANIPULANDO DADOS

Professor: Eduardo Inocencio

# VISUALIZANDO CABEÇALHOS

Podemos verificar o cabeçalho do dataframe com o comando ***df.head()***.

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown

# VISUALIZANDO AS ÚLTIMAS LINHAS

Podemos verificar o final do dataframe com o comando ***df.tail()***.

```
df.tail()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other

# NOMEANDO COLUNAS

Podemos usar o ***df.columns*** para modificar todos os nomes das colunas:

```
df.columns = [  
    'idade', 'job', 'estado civil', 'educação', 'default',  
    'balance', 'housing', 'empréstimo', 'contato', 'dia',  
    'mes', 'duracao', 'campanha', 'pdays', 'previous', 'poutcome'  
]  
df
```

Ou modificar o nome de colunas específicas usando a função ***df.rename()***

```
df.rename(columns={'idade': 'Idade', 'job': 'Profissão'})
```

	Idade	Profissão	estado civil	educação	default	balance	housing	empréstimo	contato
0	58	management	married	tertiary	no	2143	yes	no	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown

# OBTENDO INFORMAÇÕES

É essencial entendermos o tipo de dado que temos em mãos. O comando ***df.info()*** nos ajuda a verificar os tipos das nossas variáveis e, inclusive, se há valores faltantes/nulos.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 16 columns):
#   Column      Non-Null Count  Dtype
---  -
0   idade       45211 non-null  int64
1   job         45211 non-null  object
2   estado civil 45211 non-null  object
3   educação    45211 non-null  object
4   default     45211 non-null  object
5   balance     45211 non-null  int64
6   housing     45211 non-null  object
7   empréstimo  45211 non-null  object
8   contato     45211 non-null  object
9   dia         45211 non-null  int64
10  mes         45211 non-null  object
11  duracao     45211 non-null  int64
12  campanha    45211 non-null  int64
13  pdays      45211 non-null  int64
14  previous    45211 non-null  int64
15  poutcome    45211 non-null  object
dtypes: int64(7), object(9)
memory usage: 5.5+ MB
```

# ESTATÍSTICAS BÁSICAS

A função ***df.describe()*** exibe as estatísticas básicas dos dados numéricos.

```
df.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

# ESTATÍSTICAS BÁSICAS

Para estatísticas de variáveis categóricas, precisamos adicionar o argumento `include = ['O']`.

```
df.describe(include=['O'])
```

	job	marital	education	default	housing	loan	contact	month	poutcome
count	45211	45211	45211	45211	45211	45211	45211	45211	45211
unique	12	3	4	2	2	2	3	12	4
top	blue-collar	married	secondary	no	yes	no	cellular	may	unknown
freq	9732	27214	23202	44396	25130	37967	29285	13766	36959

**TOP:** O valor que mais aparece nos campos  
**FREQ:** Frequência (quantidade de vezes)  
que o valor mais comum aparece (TOP)



# ESTATÍSTICAS BÁSICAS

Ainda é possível aplicar o ***pd.describe*** a todos os dados. Para isso, precisamos adicionar o argumento ***include = 'all'***.

```
df.describe(include='all')
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	
count	45211.000000	45211	45211	45211	45211	45211.000000	45211	45211	45211	45211.000000	45211	45211.000000	45211.000000	452
unique	NaN	12	3	4	2	NaN	2	2	3	NaN	12	NaN	NaN	
top	NaN	blue-collar	married	secondary	no	NaN	yes	no	cellular	NaN	may	NaN	NaN	
freq	NaN	9732	27214	23202	44396	NaN	25130	37967	29285	NaN	13766	NaN	NaN	
mean	40.936210	NaN	NaN	NaN	NaN	1362.272058	NaN	NaN	NaN	15.806419	NaN	258.163080	2.763841	
std	10.618762	NaN	NaN	NaN	NaN	3044.765829	NaN	NaN	NaN	8.322476	NaN	257.527812	3.098021	1
min	18.000000	NaN	NaN	NaN	NaN	-8019.000000	NaN	NaN	NaN	1.000000	NaN	0.000000	1.000000	
25%	33.000000	NaN	NaN	NaN	NaN	72.000000	NaN	NaN	NaN	8.000000	NaN	103.000000	1.000000	
50%	39.000000	NaN	NaN	NaN	NaN	448.000000	NaN	NaN	NaN	16.000000	NaN	180.000000	2.000000	
75%	48.000000	NaN	NaN	NaN	NaN	1428.000000	NaN	NaN	NaN	21.000000	NaN	319.000000	3.000000	
max	95.000000	NaN	NaN	NaN	NaN	102127.000000	NaN	NaN	NaN	31.000000	NaN	4918.000000	63.000000	8

# SELECIONANDO DADOS

PODEMOS ACESSAR AS INFORMAÇÕES DOS DATAFRAMES E SERIES DE DIVERSAS MANEIRAS.

>> Quando queremos apenas 1 coluna

```
df['age']
```

0	58
1	44
2	33
3	47
4	33
..	
45206	51
45207	71
45208	72
45209	57
45210	37

Name: age, Length: 45211, dtype: int64

ou

```
df.age
```

0	58
1	44
2	33
3	47
4	33
..	
45206	51
45207	71
45208	72
45209	57
45210	37

Name: age, Length: 45211, dtype: int64

# SELECIONANDO DADOS

Podemos acessar as informações dos *dataframes* e *series* de diversas maneiras.

>> Quando queremos mais de 1 coluna

```
df[['age', 'day', 'job']]
```

	age	day	job
0	58	5	management
1	44	5	technician
2	33	5	entrepreneur
3	47	5	blue-collar
4	33	5	unknown
...	...	...	...
45206	51	17	technician
45207	71	17	retired
45208	72	17	retired
45209	57	17	blue-collar
45210	37	17	entrepreneur

# SELECIONANDO DADOS COM LOC E ILOC

Professor: Eduardo Inocencio

# SELEÇÃO - ILOC E LOC

Os métodos *iloc* e *loc* são utilizados para selecionar dados de um *dataframe*, mas possuem diferenças importantes.

>> **iloc**: seleção baseada nas posições dos índices das linhas e colunas (inteiros);

>> **loc**: seleção baseadas nos nomes das variáveis.

# SELEÇÃO - ILOC E LOC

Em ambos os casos, os argumentos dos métodos são as linhas e as colunas de interesse.

```
df.iloc [<linhas>,<colunas>]
```

```
df.loc [<linhas>,<colunas>]
```

# ILOC – SELEÇÃO DE LINHAS

O `iloc` faz a seleção através dos valores inteiros dos índices, por um array ou ainda por fatias dos dados.

```
df.iloc[1]
```

```
age          44
job          technician
marital      single
education    secondary
default      no
balance      29
housing      yes
loan         no
contact      unknown
day          5
month        may
duration     151
campaign     1
pdays      -1
previous     0
poutcome    unknown
Name: 1, dtype: object
```

```
df.iloc[[1]]
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown

Note a diferença entre os resultados. Embora os valores sejam os mesmos, a apresentação é diferente.

# ILOC – SELEÇÃO DE LINHAS

```
df.iloc[[1]]
```

Apenas uma linha.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown

```
df.iloc[[-1]]
```

Apenas uma linha.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other

```
df.iloc[0:3]
```

Um fatiamento de linhas.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown



# ILOC – SELEÇÃO DE COLUNAS

```
df.iloc[:,1]
```

Apenas uma coluna.

```
0      management
1      technician
2      entrepreneur
3      blue-collar
4      unknown
...
45206    technician
45207      retired
45208      retired
45209    blue-collar
45210    entrepreneur
Name: job, Length: 45211, dtype: object
```

```
df.iloc[:,1:3]
```

Um fatiamento de colunas.

	job	marital
0	management	married
1	technician	single
2	entrepreneur	married
3	blue-collar	married
4	unknown	single
...	...	...
45206	technician	married
45207	retired	divorced
45208	retired	married
45209	blue-collar	married
45210	entrepreneur	married

45211 rows × 2 columns

# ILOC – SELEÇÃO DE LINHAS E COLUNAS

```
df.iloc[1,0:3]
```

```
age      44  
job      technician  
marital   single  
Name: 1, dtype: object
```

Uma linha e um fatiamento de colunas.

```
df.iloc[0:2,0:3]
```

	age	job	marital
0	58	management	married
1	44	technician	single

Um fatiamento de linhas e colunas.

```
df.iloc[[0,5,8],0:3]
```

	age	job	marital
0	58	management	married
5	35	management	married
8	58	retired	married

Um array e um fatiamento de colunas.

# SELEÇÃO LOC

Selecionando valores de apenas uma linha.

```
df.loc[[1]]
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown

```
df.loc[1]
```

```
age          44
job          technician
marital      single
education    secondary
default      no
balance      29
housing      yes
loan         no
contact      unknown
day          5
month        may
duration     151
campaign     1
pdays      -1
previous     0
poutcome     unknown
Name: 1, dtype: object
```

Similar ao o que observamos no iloc.

# LOC – SELEÇÃO DE LINHAS

SELECIONANDO UMA LISTA DE LINHAS.

```
df.loc[[0,1,2]]
```

Um array de linhas.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown

# LOC – FATIANDO DADOS

```
df.loc[0:3]
```

Uma fatia de linhas.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown

# LOC – SELEÇÃO DE LINHAS E COLUNAS

SELECIONANDO UMA LISTA DE LINHAS E COLUNAS.

```
df.loc[[0,1],['age','job','default']]
```

Uma lista de linhas e uma lista de colunas.

	age	job	default
0	58	management	no
1	44	technician	no

# LOC – SELEÇÃO DE LINHAS E COLUNAS

SELECIONANDO UMA FATIA DE LINHAS E UMA LISTA DE COLUNAS.

```
df.loc[0:2,['age','job','default']]
```

Uma fatia de linhas e uma lista de colunas.

	age	job	default
0	58	management	no
1	44	technician	no
2	33	entrepreneur	no

# LOC – SELEÇÃO CONDICIONAL

SELECIONANDO COM BASE EM UMA CONDIÇÃO.

```
df.loc[df.age > 55]
```

Uma condição.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
8	58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown
13	58	technician	married	unknown	no	71	yes	no	unknown	5	may	71	1	-1	0	unknown
14	57	services	married	secondary	no	162	yes	no	unknown	5	may	174	1	-1	0	unknown
17	57	blue-collar	married	primary	no	52	yes	no	unknown	5	may	38	1	-1	0	unknown



# LOC – SELEÇÃO CONDICIONAL

SELECIONANDO UMA FATIA DE LINHAS E UMA LISTA DE COLUNAS.

```
df.loc[(df.age > 55) & (df.job == 'technician')]
```

Duas ou mais condições.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
13	58	technician	married	unknown	no	71	yes	no	unknown	5	may	71	1	-1	0	unknown
30	57	technician	married	secondary	no	839	no	yes	unknown	5	may	225	1	-1	0	unknown
35	57	technician	divorced	secondary	no	63	yes	no	unknown	5	may	242	1	-1	0	unknown
119	57	technician	married	primary	no	0	no	no	unknown	5	may	98	1	-1	0	unknown
120	56	technician	divorced	unknown	no	56	yes	no	unknown	5	may	439	1	-1	0	unknown