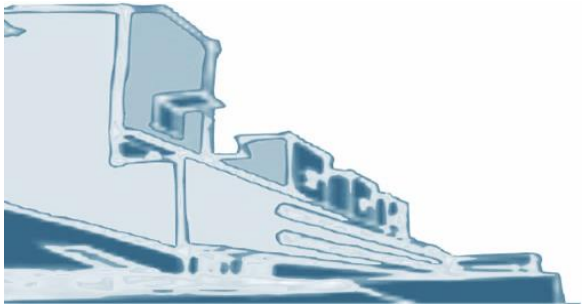
AC – Aprendizagem Computacional / Machine Learning

P0b – Pandas

Jorge Henriques
jh@dei.uc.pt

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia



1 2 9 0
UNIVERSIDADE D COIMBRA

dei engenharia informática

pandas

Introduction

- This presentation is based on the presentation of ZanoniDays
  - https://ic.unicamp.br/~mc102/aulas/aula17.pdf

Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Aula 17

Pandas

Algoritmos e Programação de Computadores

Zanoni Dias

2022

Instituto de Computação

Pandas

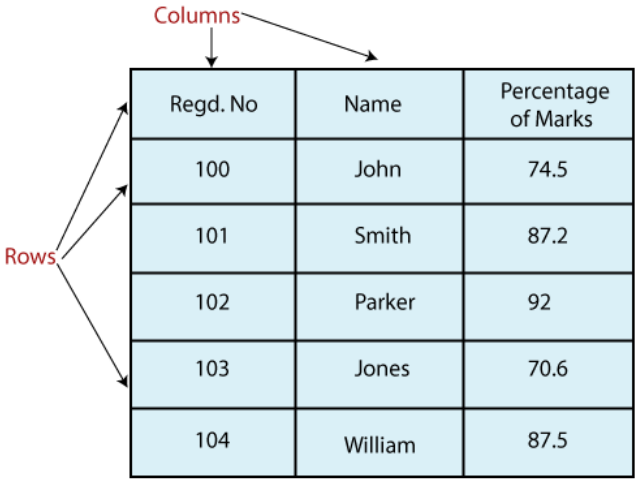
- **Introduction: series and dataFrames**
- Data Access
- Data manipulation
- Data operations
- Data import and export

Introduction

- **Pandas**
  - Pandas data structures to support the representation of structured data
    - Series
    - Dataframes

Introduction

- **Pandas**
  - open library providing functionalities to manage ***data structures*** - Python.
  - In particular *dateframes*
    - Two-dimensional data structure (table)



  - Import the library

```
» Import pandas as pd
```

Introduction

- **Series**
  - The Pandas Series object is an indexed one-dimensional (*similar to array of numpy*)

#Create an object

```
data =pd.Series([34.5, 35.7, 36.2, 36.5, 36.4])
```

Introduction

- **Series**

#List indexes, values

```
data.index
data.values
```

#access elements

```
data[1]
data[1:4]
```

Introduction

- **Series**
  - The Series object is basically similar to a vector array in NumPy
  - The difference is at the index level:
    - Array in NumPy has an integer index, implicitly defined
    - Series has an explicitly defined index, associated with the values.

```
data =pd.Series([35.5, 35.7, 36.2, 36.5, 36.4], index=['a', 'b', 'c', 'd', 'e'])
```

```
a    35.5
b    35.7
c    36.2
d    36.5
e    36.4
```

#access elements

```
data['b']
```

Introduction

- **Series:** construction of the Series object

```
pd.Series(data, index = index)
```

```
pd.Series([2, 4, 6])
```

```
0    2
1    4
2    6
```

```
pd.Series(5, index=[100, 200, 300])
```

```
100    5
200    5
300    5
```

```
pd.Series({2:'a', 1:'b', 3:'c'})
```

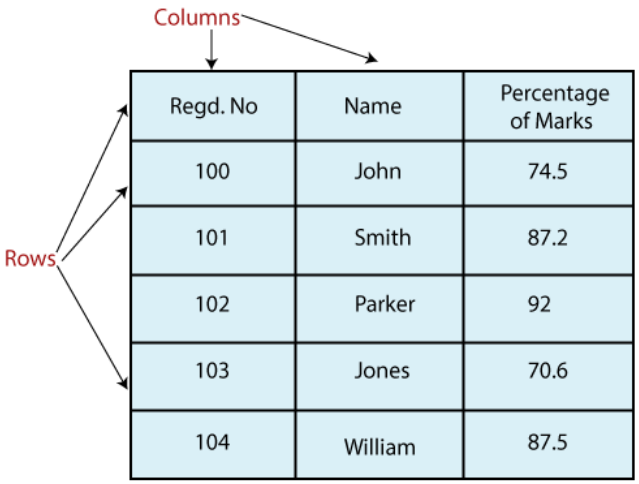
```
2    a
1    b
3    c
```

```
pd.Series({2:'a', 1:'b', 3:'c'}, index=[3, 2])
```

```
3    c
2    a
```

Introduction

- **Dataframe**
  - Two (or more) dimensional object with the following characteristics:
    - Dimensions can be modified
    - Data can be accessible through labels and indexes.
    - Use of heterogeneous data



Introduction

- Create a DataFrame from a Dictionary

```
1 import pandas as pd
2 dados = {'Nome': ['Ana', 'Bruno', 'Carla'], 'Idade': [21,
      20, 22]}
3 # {'Nome': ['Ana', 'Bruno', 'Carla'], 'Idade': [21, 20,
      22]}
4 df = pd.DataFrame(data = dados)
5 print(df)
6 #     Nome  Idade
7 # 0    Ana     21
8 # 1  Bruno     20
9 # 2  Carla     22
```

Introduction

- DataFrames allows personalized labels, to facilitate data access

```
1  import pandas as pd
2  dados = [('Ana', 21), ('Bruno', 20), ('Carla', 22)]
3  colunas = ['Nome', 'Idade']
4  linhas = ['A', 'B', 'C']
5  df = pd.DataFrame(data = dados, columns = colunas,
6                    index = linhas)
7  print(df)
8  #      Nome  Idade
9  # A     Ana     21
10 # B   Bruno     20
11 # C   Carla     22
```

Introduction

- Labels of a DataFrame can be modified after their creation: columns and index attributes

```
1  import pandas as pd
2  dados = [('Ana', 21), ('Bruno', 20), ('Carla', 22)]
3  df = pd.DataFrame(data = dados)
4  print(df)
5  #         0   1
6  # 0     Ana  21
7  # 1   Bruno  20
8  # 2   Carla  22
9  df.columns = ['Nome', 'Idade']
10 df.index = ['A', 'B', 'C']
11 print(df)
12 #      Nome  Idade
13 # A     Ana     21
14 # B   Bruno     20
15 # C   Carla     22
```

Introduction

- Attributes of an Dataframe object

index	the row labels in list format.
columns	column labels in list format.
ndim	the number of dimensions of the DataFrame
shape	the size of each of the dimensions in a tuple.
size	the number of elements (cells) of the DataFrame
empty	if the DataFrame is empty (True) or not (False).

Introduction

- Examples

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(list(df.index))
9  # ['A', 'B', 'C']
10 print(list(df.columns))
11 # ['Nome', 'Idade']
```

Introduction

- Examples

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome   Idade
5  # A     Ana      21
6  # B   Bruno      20
7  # C   Carla      22
8  print(df.ndim)
9  # 2
10 print(df.shape)
11 # (3, 2)
12 print(df.size)
13 # 6
14 print(df.empty)
15 # False
```

Access

Copy	copy of the dataFrame
head	return the first n lines of the dataFrame (default n=5)
tail	return the last n lines of the dataFrame (default n=5)

```
df2 =df.copy()
print(df2.head(n=1))
print(df2.tail(n=2))
```

Pandas

- Introduction: dateFrames
- **Data Access**
- Data manipulation
- Data Operations
- Data import and export

Access

- Data access

```
dataframe[<coluna>][<linha>]
```

- Exemplo:

```
import pandas as pd
dados = [('Ana', 21), ('Bruno', 20), ('Carla', 22)]
df = pd.DataFrame(data = dados)
print(df)
#         0   1
# 0     Ana  21
# 1   Bruno  20
# 2   Carla  22
print(df[0][0], df[0][1], df[0][2])
# Ana Bruno Carla
```

Access

T
at
iat
loc
iloc

- DataFrames have indexes for data selection

- Examples
  - T transpose
  - at access to a single element using labels
  - iat access to a single element using indexes
  - loc selection of elements using labels
  - iloc selection of elements using indexes

Access

T
at
iat
loc
iloc

- Transpose

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.T)
9  #            A       B       C
10 # Nome     Ana   Bruno   Carla
11 # Idade     21      20      22
```

Access

T
at
iat
loc
iloc

- at: access to a single element using the line and column label

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  df.at['C', 'Nome']
9  # 'Carla'
10 df.at['C', 'Idade']
11 # 22
```

Access

T
at
iat
loc
iloc

- at: indexes can not be used !
- An error will be generated

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.at['C', 'Nome'])
9  # Carla
10 print(df.at[2, 0])
11 # ValueError: At based indexing on an non-integer index can
       only have non-integer indexers
```

Access

T
at
iat
loc
iloc

- iat: access to a single element using the line and column indexes

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.iat[0, 0])
9  # Ana
10 print(df.iat[0, 1])
11 # 21
```

Access

T
at
iat
loc
iloc

- iat: labels can not be used !
- An error will be generated

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.iat[1, 0])
9  # Bruno
10 print(df.iat['B', 'Idade'])
11 # ValueError: iAt based indexing can only have integer
       indexers
```

Access

T
at
iat
loc
iloc

- loc: selection of a set of lines and columns using labels

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.loc[['A', 'C']])
9  #      Nome  Idade
10 # A     Ana     21
11 # C   Carla     22
```

Access

T
at
iat
loc
iloc

- loc: selection of a set of lines and columns using labels

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.loc[[True, False, True]])
9  #      Nome  Idade
10 # A     Ana     21
11 # C   Carla     22
12 print(df.loc[[True, False, True], 'Nome'])
13 # A       Ana
14 # C     Carla
15 # Name: Nome, dtype: object
```

Access

T
at
iat
loc
iloc

- loc: indexes can not be used !

```
1 import pandas as pd
2 ...
3 print(df)
4 #      Nome  Idade
5 # A     Ana     21
6 # B   Bruno     20
7 # C   Carla     22
8 print(df.loc[[0,2]])
9 # KeyError: "None of [Int64Index([0, 2], dtype='int64')]
      are in the [index]"
```

Access

T
at
iat
loc
iloc

- iloc: selection of a set of lines and columns using indexes

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome   Idade
5  # A     Ana      21
6  # B   Bruno      20
7  # C   Carla      22
8  print(df.iloc[[1, 2]])
9  #      Nome   Idade
10 # B   Bruno      20
11 # C   Carla      22
```

Access

T
at
iat
loc
iloc

- iloc: selection of a set of lines and columns using indexes

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  print(df.iloc[-1])
9  # Nome      Carla
10 # Idade        22
11 # Name: C, dtype: object
12 print(df.iloc[[0,2],0])
13 # A       Ana
14 # C     Carla
15 # Name: Nome, dtype: object
```

Access

T
at
iat
loc
iloc

- iloc: labels can not be used

```
1 import pandas as pd
2 ...
3 print(df)
4 #      Nome  Idade
5 # A     Ana     21
6 # B   Bruno     20
7 # C   Carla     22
8 print(df.iloc[['B', 'C']])
9 # ValueError: invalid literal for int() with base 10: 'B'
```

Pandas

- Introduction: dateframes
- Data Access
- **Data manipulation**
- Data Operations
- Data import and export

Manipulation

- add
- Modify
- Remove

- Add a new column to the dataFrame

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade
5  # A     Ana     21
6  # B   Bruno     20
7  # C   Carla     22
8  df['Sexo'] = 'F'
9  print(df)
10 #      Nome  Idade Sexo
11 # A     Ana     21    F
12 # B   Bruno     20    F
13 # C   Carla     22    F
```

Manipulation

- add
- Modify
- Remove

- Add a new column to the dataFrame using specific values

```
1  import pandas as pd
2  ...
3  print(df)
4  #      Nome  Idade Sexo
5  # A     Ana     21    F
6  # B   Bruno     20    F
7  # C   Carla     22    F
8  df['Sexo'] = ['F', 'M', 'F']
9  print(df)
10 # A     Ana     21    F
11 # B   Bruno     20    M
12 # C   Carla     22    F
```

Manipulation

- add
- Modify
- Remove

- append add at the final

```
1  import pandas as pd
2  ...
3  print(df1)
4  #      Nome  Idade Sexo
5  # A     Ana     21    F
6  # B   Bruno     20    M
7  dados = [ {'Nome': 'Carla', 'Idade': 22, 'Sexo': 'F'},
8            {'Nome': 'Daniel', 'Idade': 18, 'Sexo': 'M'} ]
9
10 df2 = df1.append(dados, ignore_index = True)
11 print(df2)
12 #      Nome  Idade Sexo
13 # 0     Ana     21    F
14 # 1   Bruno     20    M
15 # 2   Carla     22    F
16 # 3  Daniel     18    M
```

Manipulation

add
Modify
Remove

- append keeping the labels

```
1  import pandas as pd
2  ...
3  print(df1)
4  #      Nome  Idade Sexo
5  # A     Ana     21    F
6  # B   Bruno     20    M
7  dados = [ {'Nome': 'Carla', 'Idade': 22, 'Sexo': 'F'},
8            {'Nome': 'Daniel', 'Idade': 18, 'Sexo': 'M'} ]
9  df2 = pd.DataFrame(dados, index = ['D','E'])
10 df3 = df1.append(df2, ignore_index = False)
11 print(df3)
12 #      Nome  Idade Sexo
13 # A     Ana     21    F
14 # B   Bruno     20    M
15 # C   Carla     22    F
16 # D  Daniel     18    M
```

Manipulation

add
Modify
Remove

- loc modify a line

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     21    F
6  # B    Bruno     20    M
7  df.loc['B'] = ['Bento', 22, 'M']
8  df.loc['C'] = ['Carla', 22, 'F']
9  df.loc['D'] = ['Daniela', 18, 'F']
10 print(df)
11 #       Nome  Idade Sexo
12 # A      Ana     21    F
13 # B    Bento     22    M
14 # C    Carla     22    F
15 # D  Daniela     18    F
```

Manipulation

add
Modify
Remove

- iloc modify a line

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     21    F
6  # B    Bento     22    M
7  # C    Carla     22    F
8  # D  Daniela     18    F
9  df.iloc[1] = ['Bruno', 19, 'M']
10 df.iloc[3] = ['Daniel', 18, 'M']
11 print(df)
12 #       Nome  Idade Sexo
13 # A      Ana     21    F
14 # B    Bruno     19    M
15 # C    Carla     22    F
16 # D  Daniel     18    M
```

Manipulation

add
Modify
Remove

- at,iat modify elements

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     21    F
6  # B    Bruno     19    M
7  # C    Carla     22    F
8  # D   Daniel     18    M
9  df.at['C', 'Idade'] = 20
10 df.iat[0, 1] = 17
11 print(df)
12 #       Nome  Idade Sexo
13 # A      Ana     17    F
14 # B    Bruno     19    M
15 # C    Carla     20    F
16 # D   Daniel     18    M
```

Manipulation

add
Modify
Remove

drop remove lines and columns
inplace True/False to the dataFrame/to a copy

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     17    F
6  # B    Bruno     19    M
7  # C    Carla     20    F
8  # D   Daniel     18    M
9  df.drop(index = ['A', 'D'], columns = ['Sexo'],
10         inplace = True)
11 print(df)
12 #      Nome  Idade
13 # B   Bruno     19
14 # C   Carla     20
```

Pandas

- Introduction: dateframes
- Data Access
- Data manipulation
- **Operations on data**
- Data import and export

Operations

Logical and arithmetic operators

- +, +=
- -, -=
- *, *=
- /, /=
- ==, >=, <=, !=, >, <

Operations

- Add one year to all

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     17    F
6  # B    Bruno     19    M
7  # C    Carla     20    F
8  # D   Daniel     18    M
9  df['Idade'] += 1
10 print(df)
11 #       Nome  Idade Sexo
12 # A      Ana     18    F
13 # B    Bruno     20    M
14 # C    Carla     21    F
15 # D   Daniel     19    M
```

Operations

Check if Age is greater than a given threshold

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome   Idade Sexo
5  # A      Ana      17    F
6  # B    Bruno      19    M
7  # C    Carla      20    F
8  # D   Daniel      18    M
9  resultado = list(df['Idade'] >= 18)
10 print(resultado)
11 # [False, True, True, True]
```

Operations

- Select Gender=Female

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome  Idade Sexo
5  # A      Ana     17    F
6  # B    Bruno     19    M
7  # C    Carla     20    F
8  # D   Daniel     18    M
9  resultado = list(df['Sexo'] == 'F')
10 print(resultado)
11 # [True, False, True, False]
```

Operations

- Female and Age < 18

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome   Idade Sexo
5  # A      Ana      17    F
6  # B    Bruno      19    M
7  # C    Carla      20    F
8  # D   Daniel      18    M
9  resultado = list(df['Sexo'] == 'F')
10 df = df.loc[resultado]
11 resultado = list(df['Idade'] < 18)
12 print(df.loc[resultado])
13 #    Nome  Idade Sexo
14 # A   Ana      17    F
```

Operations

- Sort values

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome   Idade Sexo
5  # A      Ana      17    F
6  # B    Bruno      19    M
7  # C    Carla      20    F
8  # D   Daniel      18    M
9  df.sort_values(by = 'Idade', ascending = False,
10                 inplace = True)
11 print(df)
12 #       Nome   Idade Sexo
13 # C    Carla      20    F
14 # B    Bruno      19    M
15 # D   Daniel      18    M
16 # A      Ana      17    F
```

Operations

- Sort values using two keys

```
1  import pandas as pd
2  ...
3  print(df)
4  #       Nome   Idade Sexo
5  # A      Ana      17    F
6  # B    Bruno      19    M
7  # C    Carla      20    F
8  # D   Daniel      18    M
9  df.sort_values(by = ['Sexo', 'Idade'], inplace = True)
10 print(df)
11 #       Nome   Idade Sexo
12 # A      Ana      17    F
13 # C    Carla      20    F
14 # D   Daniel      18    M
15 # B    Bruno      19    M
```

Operations

- A biblioteca `pandas` possui vários métodos para realização de cálculos em colunas:
  - `abs`: retorna uma lista com os valores absolutos da coluna.
  - `count`: realiza a contagem de quantas células da coluna possuem valores disponíveis.
  - `nunique`: conta os valores distintos na coluna.
  - `sum`: retorna a soma dos valores da coluna.
  - `max`: retorna o maior valor da coluna.
  - `min`: retorna o menor valor da coluna.
  - `mean`: computa a média dos valores da coluna.
  - `median`: calcula a mediana dos valores da coluna.

Manipulation

- Math operations

```
import pandas as pd
print(df)
#       Nome   Idade Sexo
# A      Ana      17    F
# B    Bruno      19    M
# C    Carla      20    F
# D   Daniel      18    M
print(df.Idade.count())
# 4
print(df.Idade.sum())
# 74
print(df.Idade.max())
# 20
print(df.Idade.min())
# 17
print(df.Idade.mean())
# 18.5
print(df.Idade.median())
# 18.5
```

Pandas

- Introduction: dateframes
- Data Access
- Data manipulation
- Operation on data
- **Data import and export**

Import and export

- Import a csv file

```
1 import pandas as pd
2 df = pd.read_csv('dados.csv', index_col = 0, header = 0)
3 print(df)
4 #       Nome  Idade Sexo
5 # A      Ana     17    F
6 # B    Bruno     19    M
7 # C    Carla     20    F
8 # D   Daniel     18    M
```

Import and export

- Export a csv file

```
1 import pandas as pd
2 ...
3 print(df)
4 #       Nome  Idade Sexo
5 # A      Ana     17    F
6 # B    Bruno     19    M
7 # C    Carla     20    F
8 # D   Daniel     18    M
9 df.to_csv('dados.csv')
```