



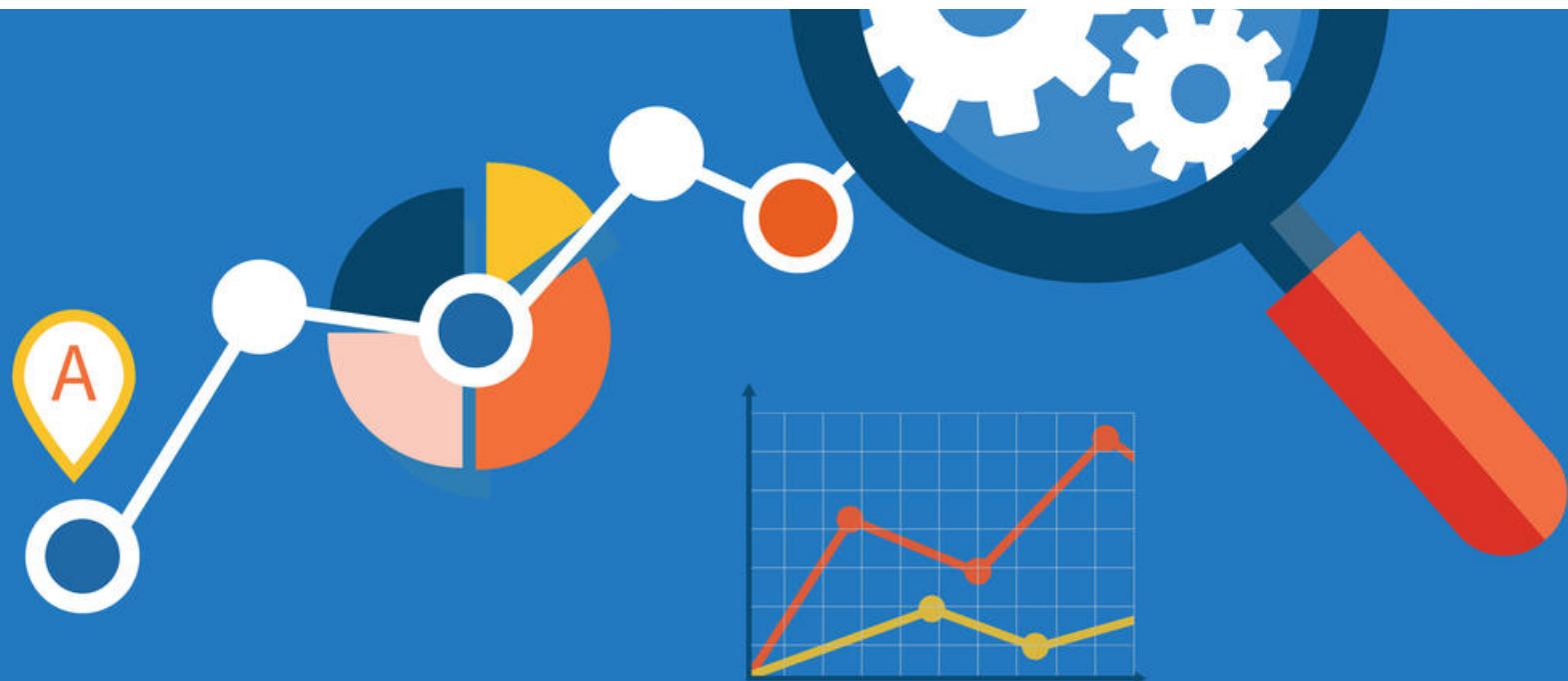
UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

INTRODUCCIÓN A PYTHON PANDAS

Background designed by kjpargeter / Freepik

Miguel Carrasco
Junio 2019

- ▶ El análisis de datos es el proceso de evaluar datos utilizando herramientas analíticas y estadísticas para descubrir información útil y ayudar en la toma de decisiones de negocios.





Pandas es un módulo (de código abierto) que proporciona:

- ▶ estructuras de datos de alto rendimiento y
- ▶ herramientas de análisis de datos

para el lenguaje de programación Python.

```
#importar pandas
import pandas as pd
```





Pandas está basado en otros módulos, tales como Numpy, y usa principalmente dos estructuras de datos:

- ▶ **Series**
- ▶ **DataFrames**

Nos enfocaremos en el último. Un DataFrame es una matriz (arreglo bidimensional) etiquetada con columnas de tipos potencialmente diferentes.

	NAME	AGE	DESIGNATION
1	a	20	VP
2	b	27	CEO
3	c	35	CFO
4	d	55	VP
5	e	18	VP
6	f	21	CEO
7	g	35	MD

¿Para qué sirve?



Un dataframe nos permite:

- Cargar datos de otra fuente de datos (como un CSV)
 - Mantener datos en forma ordenada
 - Acceder por nombre o por índice (posición)
 - Realizar operaciones sobre datos (como Numpy)
 - Entre otras posibilidades



Para aprender Pandas de forma más práctica, iremos paso a paso usando datos de los juegos olímpicos obtenidos de <https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results>

- Una copia de los datos se encuentra en webcursos para que los bajen

- ▶ Lo que haremos es:
 - Cargar los datos
 - Analizar un dataframe
 - Acceder a datos
 - Realizar filtros
 - Obtener estadísticas

Carga de Datos

```
import pandas as pd
```

```
df = pd.read_csv('athlete_events.csv')
```

read_csv() lee un archive csv
y lo carga en un dataframe

```
print(df.describe())
```

describe() nos permite conocer datos generales
del dataframe

```
print(df.head(10))
```

head(n) nos muestra los primeros n registros

				ID	Age	...
				count	271116.000000	...
				mean	68248.954396	...
				std	6.393561	...
				min	21.000000	...
				25.556898	24.000000	...
				25.556898	24.000000	...
				25.556898	28.000000	...
				25.556898	28.000000	...
				25.556898	97.000000	...
ID	Name	Sex	...	Sport	Event	Medal
0	A Dijiang	M	...	Basketball	Basketball Men's Basketball	NaN
1	A Lamusi	M	...	Judo	Judo Men's Extra-Lightweight	NaN
2	Gunnar Nielsen Aaby	M	...	Football	Football Men's Football	NaN
3	Edgar Lindenau Aabye	M	...	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	Christine Jacoba Aaftink	F	...	Speed Skating	Speed Skating Women's 500 metres	NaN

[5 rows x 15 columns]

```
import pandas as pd
```

```
df = pd.read_csv('athlete_events.csv')
```

```
print(df.iloc[0])
```

◀ Muestra el primer registro del listado de atletas

```
print(df.iloc[0,1])
```

◀ Muestra el campo 2 del primer registro (A Dijiang)

ID	1
Name	A Dijiang
Sex	M
Age	24
Height	180
Weight	80
Team	China
NOC	CHN
Games	1992 Summer
Year	1992
Season	Summer
City	Barcelona
Sport	Basketball
Event	Basketball Men's Basketball
Medal	NaN
Name:	0, dtype: object

→ df.iloc[0,1])

→ df.iloc[0,5])

```
import pandas as pd
```

```
df = pd.read_csv('athlete_events.csv')
```

```
print(df['Name'])
```

Podemos obtener todos los elementos de una columna usando su nombre

```
print(df['Name'][0])
```

Muestra el primer registro del listado anterior

0	A Dijiang
1	A Lamusi
2	Gunnar Nielsen Aaby
3	Edgar Lindenau Aabye
4	Christine Jacoba Aaftink
5	Christine Jacoba Aaftink
6	Christine Jacoba Aaftink
7	Christine Jacoba Aaftink
...	

```
import pandas as pd

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

data = df[['Name', 'Team']]  Podemos obtener más de una columna

print(data.iloc[0])  Muestra el primer registro del listado anterior
```

```
Name      A Dijiang
Team      China
Name: 0, dtype: object
```



¿Por qué cuando realizamos la consulta de 1 columna pudimos ir al primer registro haciendo [0] directamente, en cambio en la consulta con 2 columnas tuvimos que usar .iloc[0]?

- ▶ Cuando consultamos por 1 columna obtenemos una serie, el equivalente a un arreglo de 1 dimensión → funciona como una lista
- ▶ En cambio, al usar 2 o más columnas, obtenemos un dataframe → debemos usarlo como dataframe



```
import pandas as pd
```

```
df = pd.read_csv('athlete_events.csv')
```

```
print(df.shape[0])
```



shape() nos permite conocer las dimensiones... ¡igual que en Numpy!

```
unique_athletes = df['Name'].unique()
```



unique() nos permite eliminar los elementos duplicados

```
print(list(unique_athletes))
```



Para mostrar todos los datos en una colección una dimensión podemos usar list()

```
print(len(unique_athletes))
```



Al igual que Numpy, podemos usar len() para contar elementos en una colección de una dimensión

```
import pandas as pd

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

gold_winners = df[df['Medal'] == 'Gold']

unique_gold_winners = gold_winners['Name'].unique()
```



Podemos colocar condiciones para buscar en vez de usar posiciones

```
import pandas as pd

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

medal_winners = df[(df['Medal'] == 'Gold') |
(df['Medal'] == 'Silver') | (df['Medal'] == 'Bronze')]

unique_medal_winners = medal_winners['Name'].unique()
print(unique_medal_winners)
```

 *Podemos colocar multiples condiciones usando & (and), | (or) y separando con paréntesis las condiciones*

```
import pandas as pd

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

medal_winners = df[df['Medal'].fillna('None') != 'None']

unique_medal_winners = medal_winners['Name'].unique()
print(unique_medal_winners)
```



Podemos usar la función `fillna()` para darle un valor a los que no tienen medalla y luego comparar contra ese valor

```
import pandas as pd

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

opcion_1 = df[df['Medal'].fillna('None') != 'None']

opcion_2 = df[(df['Medal'] == 'Gold') | (df['Medal'] == 'Silver') | (df['Medal'] == 'Bronze')]

print(opcion_1.equals(opcion_2))
```



La función equals() permite ver si 2 dataframes tienen los mismos datos



Tiempo : 20 minutos

Los datos de los juegos olímpicos contiene las siguientes columnas:

- ID - ID único para cada atleta
- Name - El nombre del atleta
- Sex - Género (F o M)
- Age - Edad del atleta al momento de los juegos
- Height - Altura en centímetros
- Weight - Peso en kilogramos
- Team - Equipo
- NOC - National Olympic Committee – código de 3 letras
- Games - Año y temporada
- Year - Año
- Season - Temporada (Summer o Winter)
- City - Ciudad anfitriona
- Sport - Deporte
- Event - Evento
- Medal - Medalla obtenida (Gold, Silver, Bronze, o NA si no obtuvo)

Muestre los nombres (sin repetir) de los atletas para cada una de las condiciones siguientes:

1. Atletas de género femenino que hayan participado en los juegos de verano
2. Atletas de más de 190 centímetros o 90 kilos de peso



Tiempo : 20 minutos

```
import pandas as pd

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

women_summer = df[(df['Sex']=='F') & (df['Season'] == 'Summer')]
unique_women_summer = women_summer['Name'].unique()
print(unique_women_summer)

big_athletes = df[(df['Weight']>=90.0) | (df['Height'] >= 190)]
unique_big_athletes = big_athletes['Name'].unique()
print(unique_big_athletes)
```



Tiempo : 20 minutos

Más consultas

1. Muestre los nombres y deporte de los atletas de Chile que hayan participado en los juegos de 1996
2. Muestre los deportes que se han practicado en los juegos olímpicos de invierno desde 1980



Para aprender Pandas de forma más práctica, iremos paso a paso usando datos de los juegos olímpicos obtenidos de

<https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results>

- Una copia de los datos se encuentra en webcursos para que los bajen
- ▶ Lo que haremos es:
 - Cargar los datos
 - Analizar un dataframe
 - Acceder a datos
 - Realizar filtros
 - Obtener estadísticas

Agrupando datos

```
import pandas as pd
```

```
df = pd.read_csv('athlete_events.csv')
```

```
grupos = df.groupby('Year')
```



groupby () nos permite crear agrupaciones por un campo

```
print(grupos['Weight', 'Height'].agg(['min', 'max', 'mean']))
```

Year	Weight		Height			
	min	max	mean	min	max	mean
1896	45.0	106.0	71.387755	154.0	188.0	172.739130
1900	51.0	102.0	74.556962	153.0	191.0	176.637931
1904	43.0	115.0	72.197279	155.0	195.0	175.788732
1906	52.0	114.0	75.917073	165.0	196.0	178.206226
1908	51.0	115.0	75.386128	157.0	201.0	177.543158
1912	49.0	125.0	73.117450	157.0	200.0	177.447989
1920	48.0	146.0	73.106157	142.0	197.0	175.752282
1924	44.0	146.0	71.678261	142.0	200.0	174.963039
1928	41.0	125.0	70.987465	147.0	211.0	175.162051
1932	41.0	110.0	70.544545	147.0	200.0	174.220115
1936	37.0	138.0	71.505572	147.0	205.0	175.723993
1948	47.0	125.0	71.614248	140.0	213.0	176.172797
1952	42.0	145.0	70.034522	150.0	213.0	174.138940
1956	28.0	141.0	70.388433	137.0	218.0	173.900968
1960	36.0	141.0	69.315195	137.0	218.0	173.141286...



Y sobre esos grupos podemos sacar estadísticas de ciertos campos, en este caso de peso y altura



Tiempo : 10 minutos

1. Compare el peso y altura promedio de los atletas de invierno con los de verano.
Para ello, primero debe agrupar los atletas para cada temporada, y luego seleccionar altura y peso y pedir los valores promedio solamente (mean)
2. Ahora realice la comparación anterior, pero solamente para los juegos olímpicos desde 1996 a la fecha

Recuerde que tiene los siguientes campos:

- Height - Altura en centímetros
- Weight - Peso en kilogramos
- Season - Temporada (Summer o Winter)



Tiempo : 10 minutos

```
import pandas as pd

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

grupos = df.groupby('Season')
print(grupos['Weight', 'Height'].agg(['mean']))

desde_1996 = df[df['Year']>=1996].groupby('Season')
print(desde_1996['Weight', 'Height'].agg(['mean']))
```

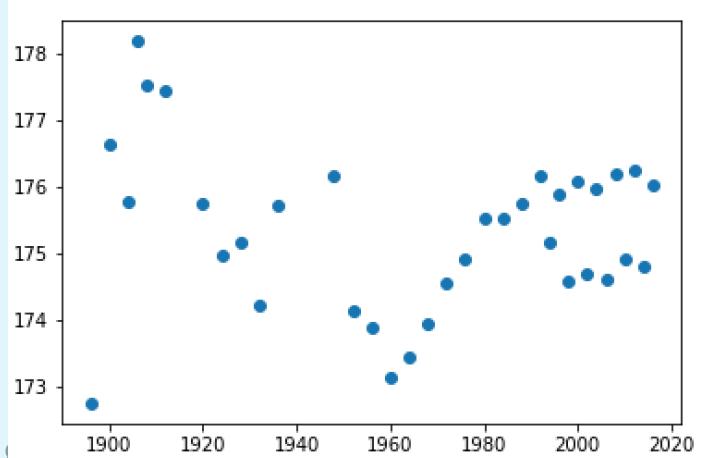
Graficando datos

```
import pandas as pd
import matplotlib.pyplot as plt

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

grupos = df.groupby('Year')
datos = grupos['Weight', 'Height'].agg(['mean'])

plt.scatter(datos.index, datos ['Weight']['mean'])
```



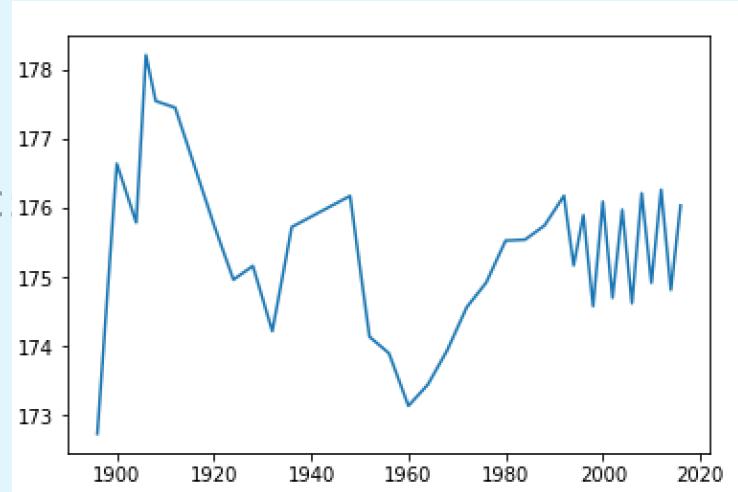
scatter() nos permite graficar puntos, en este caso usamos index, y la media de los pesos

Graficando datos

```
import pandas as pd
import matplotlib.pyplot as plt

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

grupos = df.groupby('Year')
datos = grupos['Weight', 'Height'].mean()
```



```
plt.scatter(datos.index, datos ['Weight']['mean'])
plt.plot(datos.index, datos ['Height']['mean'])
```



plot () nos permite graficar puntos, en este caso usamos index, y la media de las alturas

```
import pandas as pd
import matplotlib.pyplot as plt

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

grupos = df.groupby('Year')
datos = grupos['Weight', 'Height'].agg(['min', 'max',
                                         'mean'])

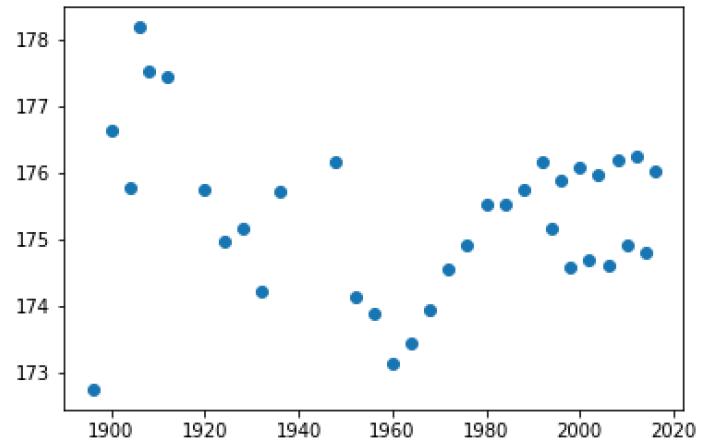
plt.scatter(datos.index, datos ['Weight']['mean'])
plt.plot(datos.index, datos ['Weight']['mean'])
plt.savefig('grafico.png')
```



savefig() permite guardar el gráfico en una imagen

Inspeccionando datos

- ▶ Observe el gráfico que obtuvimos ¿Qué puede deducir de este gráfico?



mpl.Scatter



mpl.Plot

```
import pandas as pd
import matplotlib.pyplot as plt

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

mujeres_2016 = df[(df['Sex']=='F') & (df['Year'] == 2016)]
alturas = mujeres_2016['Height'].dropna()


dropna() elimina los registros que no tienen dato de Altura



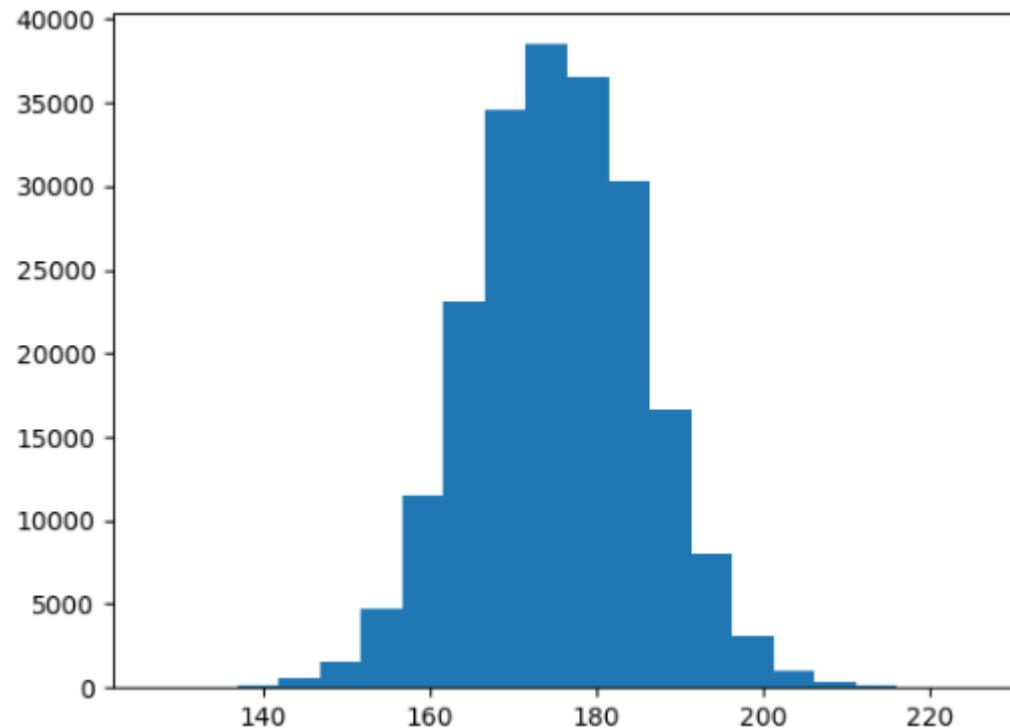
plt.hist(alturas, bins=20)


hist() nos permite graficar histogramas (agrupación por rango de Alturas en este caso)



plt.savefig('histograma.png')
```

- ▶ Observe el gráfico que obtuvimos ¿Qué puede deducir de este gráfico?





Tiempo : 30 minutos

1. Liste todos los deportes de los juegos olímpicos de verano
2. Seleccione un deporte y analice los datos de los atletas en el tiempo:
 - Peso
 - Altura
 - Edad
3. Grafique los datos que juzgue relevantes