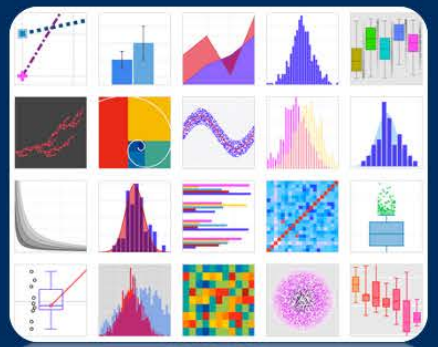
Introducción al Análisis y Visualización de Datos en Python

Día 3 – Manipulación de Datos



NumPy SciPy matplotlib

Pandas

Presentan:

Dr. Ulises Olivares Pinto

Walter André Rosales Reyes

Escuela Nacional de Estudios Superiores Unidad Juriquilla

Universidad Nacional Autónoma de México

UNAM La Universidad de la Nación

ENES Juriquilla

Contenido

1. Manipulación de datos

(~3 horas - 2 bloques)

Importación de datos CSV y TXT

Pandas

- Dataframes
- Ordenamientos
- Subconjuntos de filas y columna
- Unión de dataframes
- Estadísticos Básicos

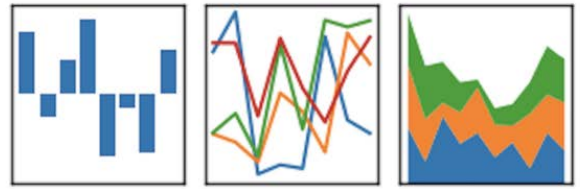
2. Proyecto Pandas

(30 ~45 minutos)

Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

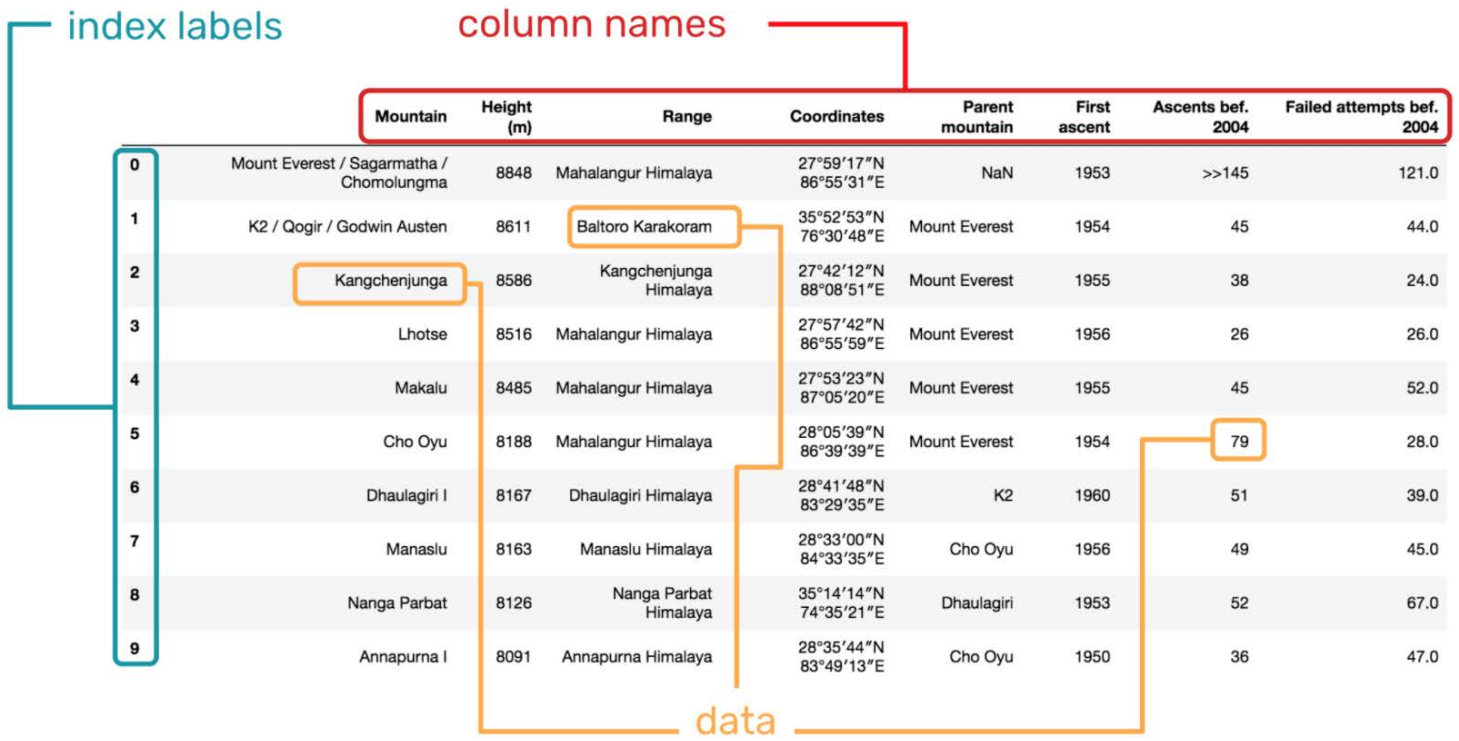


Pandas

Data frames

Es una colección de datos habitualmente tabulada.

Contenida en una única tabla cada columna representa una variable y cada fila representa un registro.



	Mountain	Height (m)	Range	Coordinates	Parent mountain	First ascent	Ascents bef. 2004	Failed attempts bef. 2004
0	Mount Everest / Sagarmatha / Chomolungma	8848	Mahalangur Himalaya	27°59′17″N 86°55′31″E	NaN	1953	>>145	121.0
1	K2 / Qogir / Godwin Austen	8611	Baltoro Karakoram	35°52′53″N 76°30′48″E	Mount Everest	1954	45	44.0
2	Kangchenjunga	8586	Kangchenjunga Himalaya	27°42′12″N 88°08′51″E	Mount Everest	1955	38	24.0
3	Lhotse	8516	Mahalangur Himalaya	27°57′42″N 86°55′59″E	Mount Everest	1956	26	26.0
4	Makalu	8485	Mahalangur Himalaya	27°53′23″N 87°05′20″E	Mount Everest	1955	45	52.0
5	Cho Oyu	8188	Mahalangur Himalaya	28°05′39″N 86°39′39″E	Mount Everest	1954	79	28.0
6	Dhaulagiri I	8167	Dhaulagiri Himalaya	28°41′48″N 83°29′35″E	K2	1960	51	39.0
7	Manaslu	8163	Manaslu Himalaya	28°33′00″N 84°33′35″E	Cho Oyu	1956	49	45.0
8	Nanga Parbat	8126	Nanga Parbat Himalaya	35°14′14″N 74°35′21″E	Dhaulagiri	1953	52	67.0
9	Annapurna I	8091	Annapurna Himalaya	28°35′44″N 83°49′13″E	Cho Oyu	1950	36	47.0

Lectura de datos con Pandas

```
In [ ]: #Leer archivo CSV
        df = pd.read_csv("https://raw.githubusercontent.com/ulises1229/INTRO-PYTHON-
                ENESJ/master/data/salaries.csv")
```

Nota: El comando anterior tiene muchos argumentos opcionales para ajustar el proceso de importación de datos.

Hay una serie de commandos en pandas para leer **otros formatos de datos**:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Explorar data frames

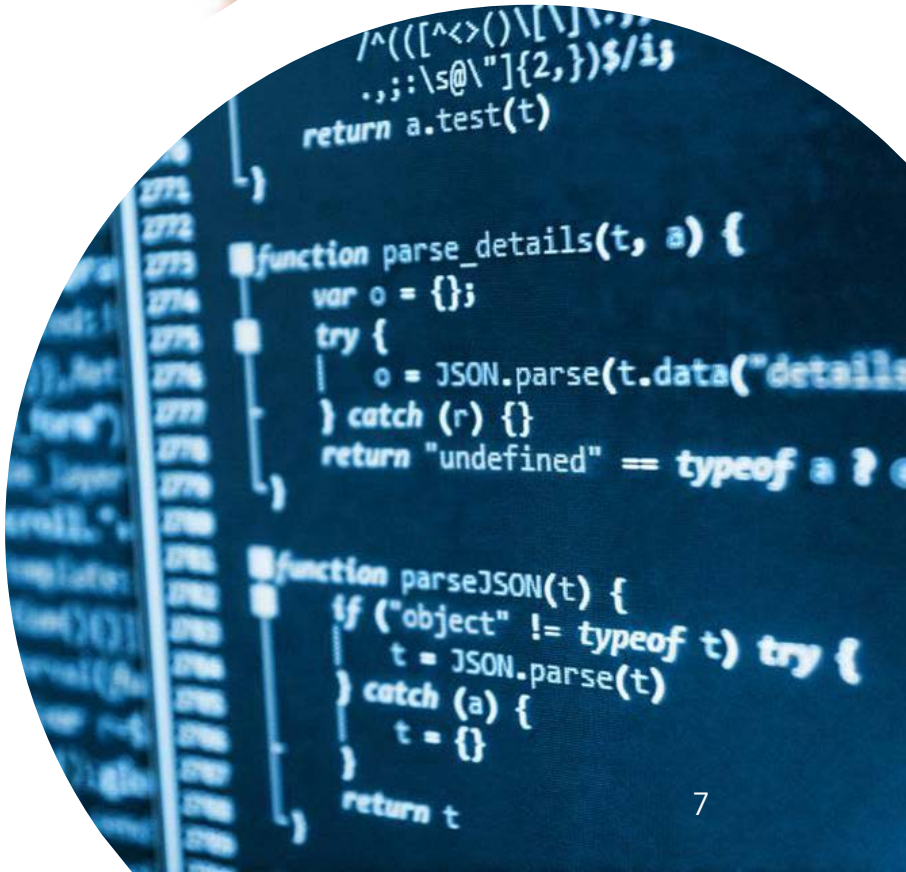
In [3]:
```
#Lista de los primeros 5 registros
df.head()
```

Out[3]:

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Ejercicio 1

- Se deberán leer los primeros 10, 20, 50 registros;
- ¿Podrían adivinar cómo ver los últimos registros;
- Pista:



Data Frames: Tipos de datos

Tipos en pandas	Tipos nativos en Python	Descripción
object	string	El dtype más general. Se asignará a la columna si la columna tiene tipos mixtos (números y cadenas).
int64	int	Caracteres numéricos. 64 se refiere a la memoria asignada para contener este carácter.
float64	float	Caracteres numéricos con decimales. Si una columna contiene números y NaNs (ver abajo), pandas usará por defecto float64, en caso de que el valor que falta tenga un decimal.
datetime64, timedelta[ns]	N/A (Ver el módulo datetime en Python)	Valores destinados a contener datos de tiempo. Busque en estos experimentos de series temporales.

Data Frame: Tipos de datos

```
In [4]: # Comprobar un tipo de columna determinado
        df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #Comprueba los tipos de todas las columnas
        df.dtypes
```

```
Out[4]: rank          object
        discipline    object
        phd           int64
        service       int64
        sex           object
        salary        int64
        dtype: object
```

Atributos de Data Frames

Los objetos de Python tienen atributos y métodos.

Atributos: usualmente se acceden a través de .

Métodos: usualmente se acceden a través de ( )

df.attribute	Descripción
dtypes	enumerar los tipos de las columnas
columns	listar los nombres de columna
axes	listar las etiquetas de fila y los nombres de columna
ndim	número de dimensiones
size	número de elementos
shape	devolver una tupla que representa la dimensionalidad
values	representación numpy de los datos

Ejercicio 2

1. Encuentre cuántos registros tiene este DataFrame
2. ¿Cuántos elementos hay?
3. ¿Cuáles son los nombres de columna?
4. ¿Qué tipos de columnas tenemos en este marco de datos?

Métodos para Data Frames

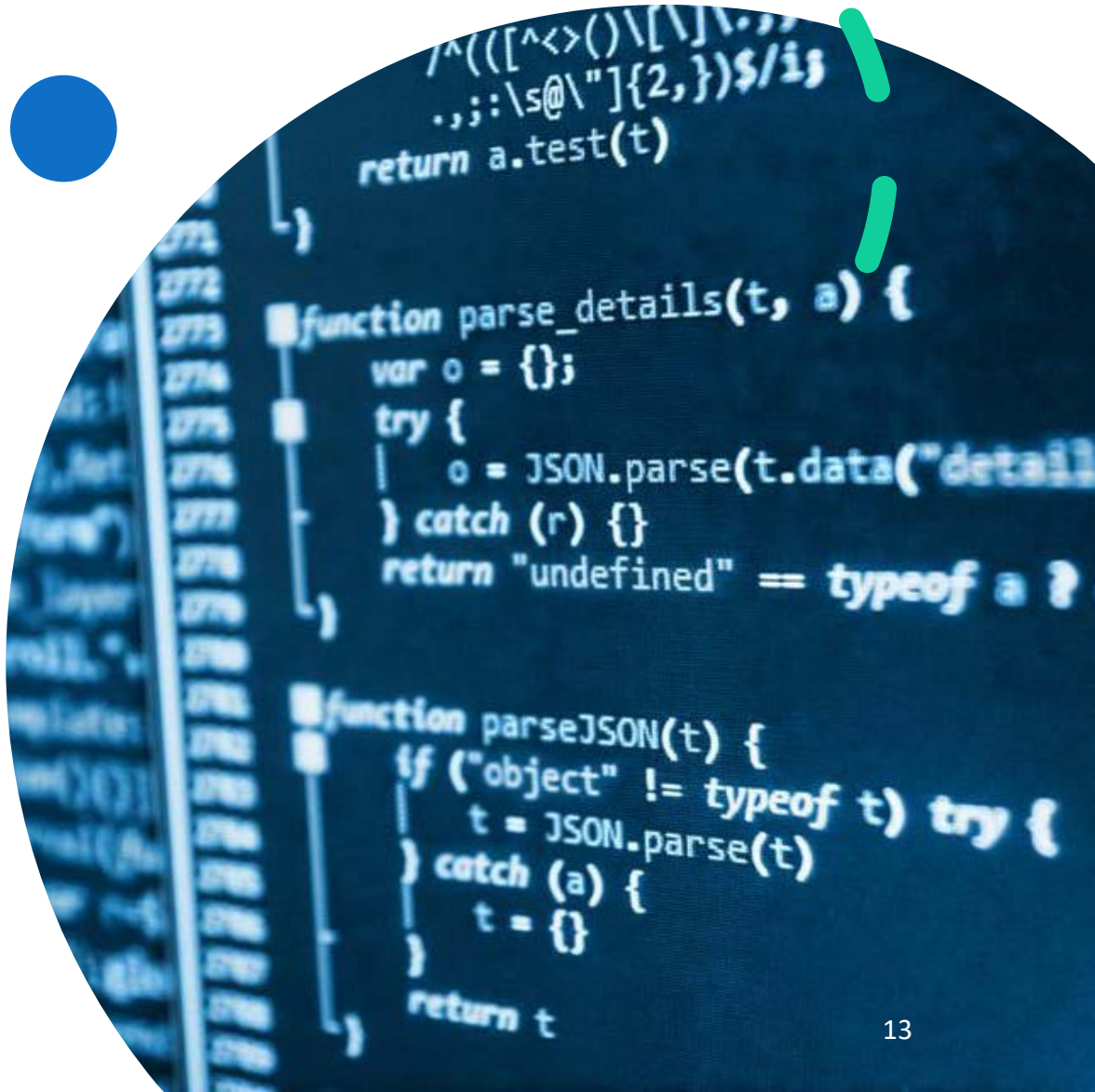
A diferencia de los atributos, los métodos en Python tienen paréntesis.

Todos los atributos y métodos se pueden enumerar con una función dir(): dir(df)

df.method()	Descripción
head([n]), tail([n])	Primera/última n filas
describe()	Generar estadísticas descriptivas (solo para columnas numéricas)
max(), min()	Devolver valores máximos/min para todas las columnas numéricas
mean(), median()	Devolver la media/mediana para todas las columnas numéricas
std()	Desviación estándar
sample([n])	devuelve una muestra aleatoria del marco de datos
dropna()	Eliminar todos los registros con valores NA

Ejercicio 3

1. Proporcione el resumen de las columnas numéricas del conjunto de datos
2. Calcular la desviación estándar para todas las columnas numéricas;
3. ¿Cuáles son las medias de los primeros 50 registros del conjunto de datos?
   - ***Sugerencia:*** *Utilice el método head() para crear un subconjunto de los primeros 50 registros y, a continuación, calcule la media*



Selección de una columna en el Data Frame

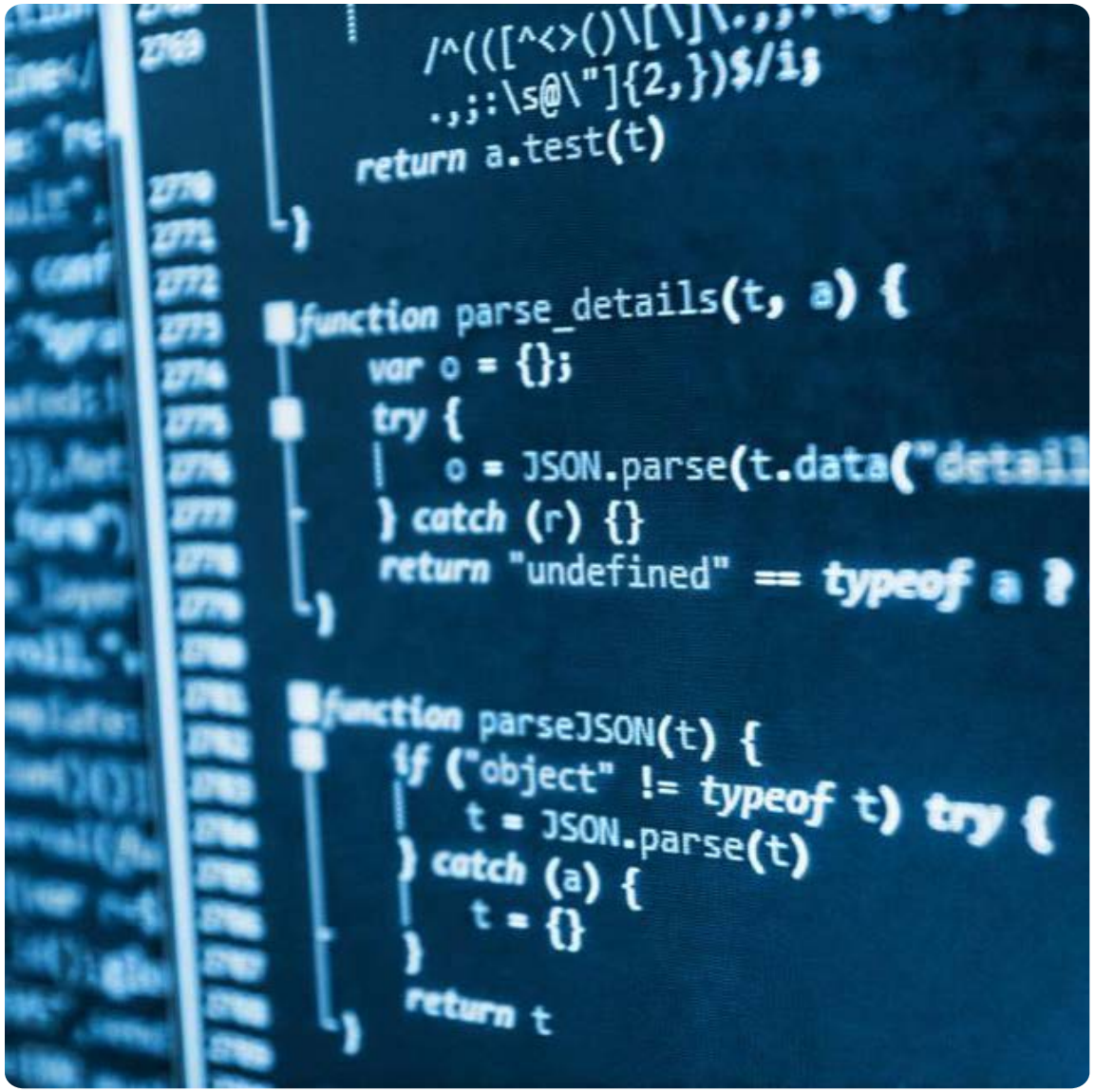
Método 1: *Subconjunto del DF utilizando el nombre de columna:*

df['sex']

Método 2: *Utilice el nombre de columna como atributo:*

df.sex

Nota: Existe un atributo rank para los data frames de pandas, por lo que para seleccionar una columna con un nombre "rank" debemos usar el método 1.



Ejercicio 4

1. Calcular las estadísticas básicas para la columna de salario;
2. Encuentre cuántos valores hay en la columna de salario (utilice el método count);
3. Calcular la media para la variable salario;

Método *groupby* para Data Frames

Usando el método **"group by"** podemos:

- Dividir los datos en grupos en función de algunos criterios
- Calcular estadísticas (o aplicar una función) a cada grupo
- Similar a la función dplyr() en R

```
In [ ]: #Agrupando los datos usando rank
        df_rank = df.groupby(['rank'])
```

```
In [ ]: #Calcular la media de cada
        columna numérica por cada grupo
        df_rank.mean()
```

rank	phd	service	salary
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Método *groupby* para Data Frames

Una vez creado el objeto groupby podemos calcular varias estadísticas para cada grupo:

In [ ]:
```
#Calculate mean salary for each professor rank:
df.groupby('rank')[['salary']].mean()
```

	salary
rank	
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Nota: Si se utilizan corchetes individuales para especificar la columna (por ejemplo, salario), la salida es el objeto Pandas Series. Cuando se utilizan corchetes dobles, la salida es un data frame

Método *groupby* para Data Frames

- ***groupby*** *notas de rendimiento:*
  - *No se produce ninguna agrupación/división hasta que sea necesario.*
  - *La creación del objeto groupby solo comprueba que ha pasado una asignación válida*
  - *Por defecto las claves de grupo se ordenan durante la operación groupby. Se puede que desee pasar sort-False para una posible aceleración:*

```
In [ ]: #Calculate mean salary for each professor rank:
        df.groupby(['rank'], sort=False)[['salary']].mean()
```

Data Frame: filtering

Para crear un subconjunto de los datos, podemos aplicar la indexación booleana. Esta indexación se conoce comúnmente como filtro. Por ejemplo, si queremos crear un subconjunto de las filas en las que el valor salarial es mayor que $120K:

In [ ]:
```
#Calcular el salario promedio para cada rango de profesor:
df_sub = df[ df['salary'] > 120000 ]
```

Cualquier operador booleano se puede utilizar para crear un subconjunto de los datos:

In [ ]:
```
#Select only those rows that contain female professors:
df_f = df[ df['sex'] == 'Female' ]
```

Data Frames: Cortes

Hay varias maneras de crear un subconjunto en un data frame:

- Una o más columnas
- Una o más filas
- Un subconjunto de filas y/o de columnas

Las filas y columnas se pueden seleccionar por su posición o etiqueta

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2
7	Qn1	Quebec	nonchilled	1000	39.7
8	Qn2	Quebec	nonchilled	95	13.6
9	Qn2	Quebec	nonchilled	175	27.3
10	Qn2	Quebec	nonchilled	250	37.1

Data Frames: Conjuntos

Al seleccionar una columna, es posible utilizar un único conjunto de corchetes, pero el objeto resultante será una serie (no un data frame):

In [ ]:
```
#Seleccione la columna de  salario:
df['salary']
```

Cuando necesitamos seleccionar más de una columna y/o hacer que la salida sea un Data Frame, debemos usar corchetes dobles:

In [ ]:
```
#Select column salary:
df[['rank','salary']]
```

Data Frames: Selección de filas

Si necesitamos seleccionar un rango de filas, podemos especificar el rango usando ":"

In [ ]:
```
#Seleccionar filas por su posición:
df[10:20]
```

Observe que la primera fila tiene una posición 0 y se omite el último valor del intervalo:

Así que para el rango 0:10 las primeras 10 filas se devuelven con las posiciones que comienzan con 0 y terminan con 9

Data Frames: método loc

Si necesitamos seleccionar un rango de filas, usando sus etiquetas podemos usar el método loc:

In [ ]:
```
#Seleccione filas por sus etiquetas:
df_sub.loc[10:20,['rank','sex','salary']]
```

Out[ ]:

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Data Frames: método iloc

Si necesitamos seleccionar un rango de filas y/o columnas, usando sus posiciones podemos usar el método iloc:

In [ ]:
```
#Select rows by their labels:
df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

Data Frames: Método iloc (resumen)

```
df.iloc[0]  # Primera fila de un marco de datos
df.iloc[i]  #(i+1)fila
df.iloc[-1] # Última fila
```

```
df.iloc[:, 0]  # Primera columna
df.iloc[:, -1] # Última columna
```

```
df.iloc[0:7]        #Primeras 7 filas
df.iloc[:, 0:2]     #Primeras 2 columnas
df.iloc[1:3, 0:2]   #De la segunda a la tercer fila y las primeras 2 columnas
df.iloc[[0,5], [1,3]]  #1a y 6a fila y 2a y 4a columnas
```

Ejercicio 5

1. Se deberá utilizar iloc para mostrar la primera y la última fila.
2. Se deberá calcular el promedio del salario de las filas [10:30]
3. Se deberá contar el número de hombres en todo el data frame.
4. Se deberán contar el número de profesores asociados (AssocProf) en las filas [50:70]

Data Frames: Ordenamientos

Podemos ordenar los datos por un valor en la columna. De forma predeterminada, la ordenación se producirá en orden ascendente y se devolverá un nuevo marco de datos.

In [ ]:
```
# Cree un nuevo marco de datos a partir del original ordenado por la columna Salario
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Data Frames: Ordenamiento

Podemos ordenar los datos usando 2 o más columnas:

In [ ]:
```
df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False])
df_sorted.head(10)
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

Ejercicio 5

1. Se deberán recuperar el profesor mejor y peor pagado de todo el dataframe.
2. Se deberá realizar el ejercicio anterior por sexo.
3. Se deberá calcular un promedio del salario por sexo.

Valores faltantes

Los valores que faltan se marcan como NaN

In [ ]:
```
# Leer un conjunto de datos con valores faltantes
vuleos = pd.read_csv(" https://raw.githubusercontent.com/ulises1229/INTRO-PYTHON-ENESJ/master/data/flights.csv ")
```

In [ ]:
```
# Seleccione las filas que tienen al menos un valor faltante
vuelos[vuelos.isnull().any(axis=1)].head()
```

Out[ ]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N31412	1228	EWR	SAN	NaN	2425	18.0	7.0
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EHAA	791	LGA	DFW	NaN	1389	NaN	NaN
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EVAA	1925	LGA	MIA	NaN	1096	NaN	NaN
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N12221	1299	EWR	RSW	NaN	1068	21.0	45.0
858	2013	1	2	NaN	NaN	NaN	NaN	AA	NaN	133	JFK	LAX	NaN	2475	NaN	NaN

Valores faltantes

Hay una serie de métodos para tratar con los valores que faltan en el marco de datos:

df.method()	Descripción
dropna()	Retirar observaciones faltantes
dropna(how='all')	Elimina todas las celdas con NA
dropna(axis=1, how='all')	Elimina la columna si faltan todos los valores
dropna(thresh = 5)	Elimina las filas que contienen menos de 5 valores que no faltan
fillna(0)	Reemplaza los valores faltantes por ceros
isnull()	Devuelve True si falta el valor
notnull()	Devuelve True para los valores que no faltan

Valores faltantes – Características

- Al sumar los datos, los valores que faltan se tratarán como cero
- Si faltan todos los valores, la suma será igual a NaN
- Los métodos cumsum() y cumprod() ignoran los valores faltantes, pero los preservan en los arrays resultantes
- Los valores faltantes se excluyen en método GroupBy (al igual que en R)
- Muchos métodos estadísticos descriptivos tienen la opción skipna para controlar si se deben excluir los datos que faltan. Este valor se establece en True de forma predeterminada (a diferencia de R)

Funciones de agregación en pandas

Agregación - calcular una estadística de resumen sobre cada grupo, es decir.

- Calcular grupos sumas o promedios

- Funciones comunes de agregación:
  - min, max
  - count, sum, prod
  - mean, median, mode, mad
  - std, var

Funciones de agregación en pandas

El método agg() es útil cuando se calculan varias estadísticas por columna:

In [ ]:
```
Vuelos[['dep_delay','arr_delay']].agg(['min','mean','max'])
```

Out[ ]:

	dep_delay	arr_delay
min	-16.000000	-62.000000
mean	9.384302	2.298675
max	351.000000	389.000000

Estadísticas descriptivas básicas

df.method()	Descripción
describe	Estadísticas básicas (count, mean, std, min, quantiles, max)
min, max	Valores mínimos y máximos
mean, median, mode	Promedio aritmético, mediana y moda
var, std	Variación y desviación estándar
sem	Error estándar de media
skew	Asimetría de la muestra
kurt	Curtosis

Gráficos para explorar los datos

El paquete Seaborn se basa en matplotlib, pero proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos, similar a la biblioteca ggplot2 en R. Se dirige específicamente a la visualización de datos estadísticos

Para mostrar gráficos dentro del bloc de notas de Python, incluya la directiva en línea:

```
In [ ]: %matplotlib inline
```

Gráficos

Gráfico	Descripción
distplot	Histograma
barplot	Tendencia central a una variable numérica
violinplot	Similar a la gráfica de caja, también muestra la densidad de probabilidad de los datos
jointplot	Scatterplot
regplot	Gráfica de regresión
pairplot	Pairplot
boxplot	Boxplot
swarmplot	Gráfica de dispersión categórica
factorplot	Gráfico general categórico

Análisis estadístico básico

statsmodel y scikit-learn - ambos tienen una serie de funciones para el análisis estadístico

El primero se utiliza principalmente para el análisis regular utilizando fórmulas de estilo R, mientras que scikit-learn es más personalizado para Machine Learning.

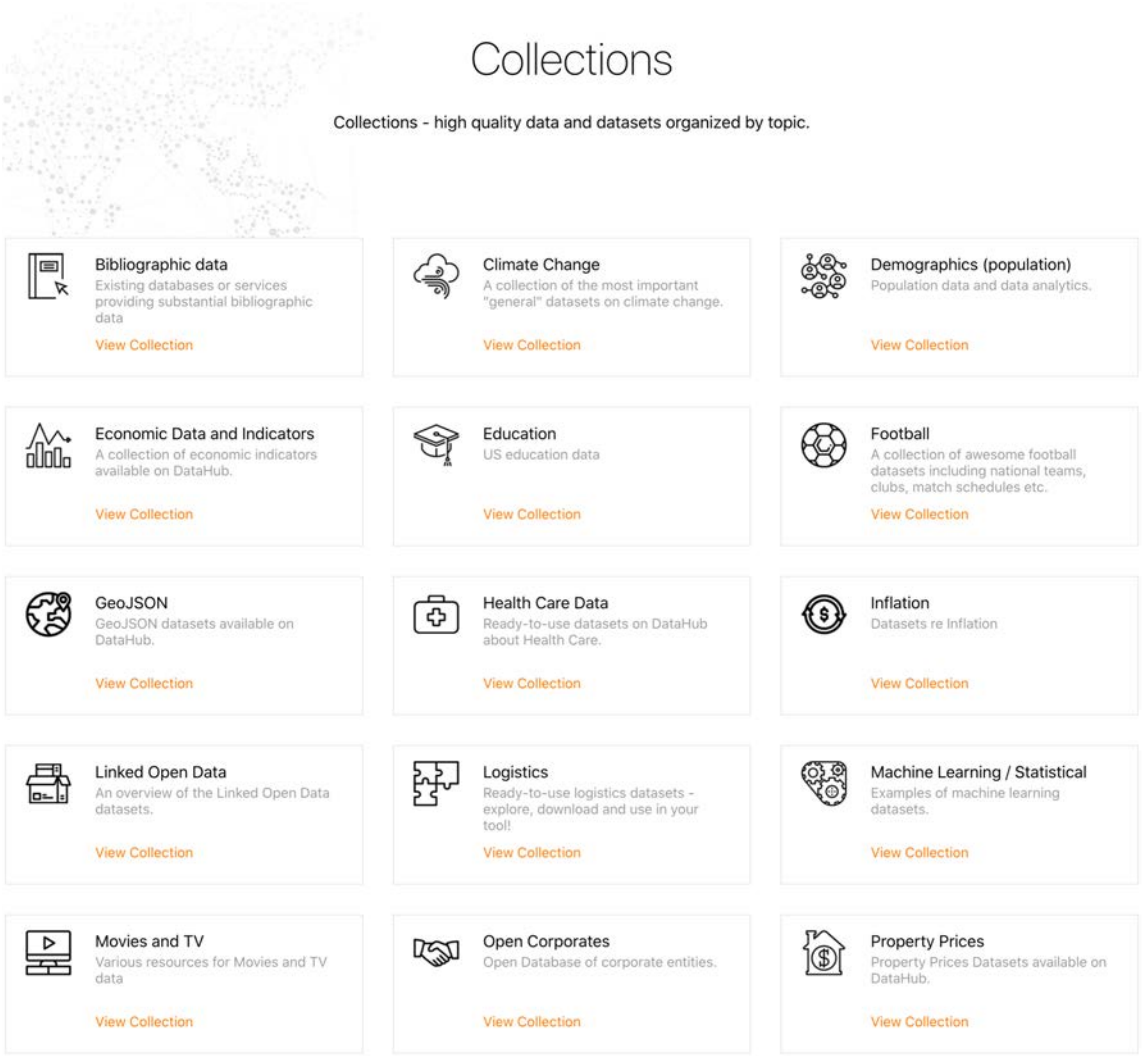
statsmodels:

- Regresiones lineales
- Pruebas ANOVA
- Prueba de hipótesis
- Muchas más ...

scikit-learn:

- kmeans
- support vector machines
- random forests

Proyecto Alternativo



Ejercicio Alternativo de Entrega

- De forma individual, deberán seleccionar una **categoría** y **set de datos** en la plataforma datahub: https://datahub.io/.

1. Deberán obtener al menos 3 métricas de estadística descriptiva de todo el conjunto de datos
   - **Media**
   - **STD**
   - **MEDIANA**
   - **MODA**
   - **VAR**
2. Repetir el ejercicio anterior solo para un subconjuto de datos.