*Clase 06 - Introducción a la manipulación de datos con Pandas (Parte II)*

# Agregaciones con Pandas

Pandas permite realizar agregaciones, al igual que vimos con los objetos NumPy. Aquí el concepto se vuelve más flexible, especialmente al operar con objetos DataFrame. Recordemos que las agregaciones son operaciones que obtienen como resultado medidas resumen sobre los datos. Veamos un ejemplo a continuación.

Primero vamos a crear el DataFrame que vamos a usar a partir de un archivo de origen csv. El comando pd.read\_csv(ruta de acceso al archivo) lee un archivo CSV desde nuestros archivos. CSV es un tipo de archivo separado por comas (Comma Separated Values) que tiene la forma de una hoja de cálculo, pero es más flexible porque puede abrirse directamente con un editor de texto. Aquí asumiremos que ya tenemos los datos disponibles para nuestros ejemplos, y trabajaremos desde ese punto de partida.

| # Descargar el archivo pune\_1965\_to\_2002.csv # https://www.kaggle.com/abhishekmamidi/precipitation-data-of-pune-from-1965-to-2002 df\_lluvias\_archivo = pd.read\_csv('/Users/guillermoleale/Downloads/pune\_1965\_to\_2002.csv') # Leer el archivo y guardarlo en un DataFrame indice = list(df\_lluvias\_archivo.Year) # Quiero cambiar los índices por el año, entonces descompongo el archivo en su índice, columnas y valores columnas = df\_lluvias\_archivo.columns[1:] # La columna original de año ya no es más columna sino índice valores = df\_lluvias\_archivo.values[:,1:]  df\_lluvias = pd.DataFrame(valores,index=indice,columns=columnas) df\_lluvias |
| --- |

Una vez generado el objeto DataFrame con el que vamos a trabajar, realicemos algunas operaciones sobre él:

| print(df\_lluvias.sum()) # Suma de lluvias por mes print(df\_lluvias.mean(axis='columns')) # Promedio de lluvias por año print(df\_lluvias.describe()) # Resumen estadístico por mes print(df\_lluvias.T.describe()) # Resumen estadístico por año |
| --- |

De esta forma podemos ver algunos de los comandos de agregaciones de Pandas, que hacen muy práctico el análisis de datos.

# Operaciones con strings

Un string, o cadena de caracteres, es un objeto de texto de longitud arbitraria. Las operaciones con strings son particulares, y en muchos casos diferentes a las que se realizan con números. Ejemplos de estas operaciones son ordenar alfabéticamente, pasar a mayúsculas o minúsculas, separar palabras por espacios u otro caracter especial, etcétera.

Estas operaciones son muy útiles en la etapa de Data Wrangling, y si bien Python maneja este tipo de datos, Pandas provee un conjunto de funciones que extienden y facilitan las operaciones nativas. Veamos entonces algunos ejemplos en código:

| # Descargue el archivo de presidentes en el siguiente enlace  # <https://www.kaggle.com/harshitagpt/us-presidents>  Presidentes\_archivo = pd.read\_csv('/Users/guillermoleale/Downloads/us\_presidents.csv') Presidentes\_nombres = pd.Series(Presidentes\_archivo['president']) print(Presidentes\_nombres) # Nombres de todos los presidentes de EEUU print(Presidentes\_nombres.str.upper()) # Todos los nombres en mayúsculas print(Presidentes\_nombres.str.len()) # Longitud de cada nombre, incluyendo puntos y espacios print(Presidentes\_nombres.str.startswith('J')) # ¿Qué nombres completos tienen "J" como primera letra? print(Presidentes\_nombres.str.split()) # Separar cada texto por espacios. Ojo, presidentes con segundo nombre generan tres strings. |
| --- |

Estas son sólo algunas de las poderosas operaciones que ofrece Pandas con objetos de tipo String. La lista oficial de funciones con ejemplos está disponible en el sitio https://pandas.pydata.org/pandas-docs/stable/user\_guide/text.html#string-methods

# Introducción a Series de Tiempo

Las Series de tiempo son tipos de datos especiales donde el tiempo toma un rol fundamental. En particular los datos se analizan a través del tiempo, por lo cual el índice de los datos necesariamente está dado por algún tipo de dato que tiene que ver con una unidad de tiempo.

En Python, se definen tres tipos de datos relacionados con el tiempo:

* El **Time stamp**, o “marca de tiempo”, que representa un punto en el tiempo (por ejemplo una fecha y hora determinadas)
* El **Período**, que representa un intervalo de tiempo entre un momento puntual de inicio y un momento de finalización. Por ejemplo los días transcurridos entre el 10 de enero y el 15 de enero de 2020.
* La **Duración**, que representa precisamente una duración medida en tiempo, independiente del momento en que sucede. Por ejemplo, 15 minutos.

Pandas define, para cada uno de los tres tipos mencionados anteriormente, un tipo de datos de índice (index), el cual puede usarse posteriormente para servir de elemento índice en la construcción de un objeto Series o DataFrame. Los índices para cada tipo de dato son los siguientes:

| Tipo de dato | Objeto en Python | Índice en Pandas |
| --- | --- | --- |
| Time stamp | Timestamp | DatetimeIndex |
| Período | Period | PeriodIndex |
| Duración | Timedelta | TimedeltaIndex |

Veamos algunos ejemplos de operaciones con índices de tiempo:

| fecha = pd.to\_datetime('03/01/2020',dayfirst=True) # Definiendo una fecha a partir de un texto. Si omitimos dayfirst, la fecha por defecto es mes/día/año print(fecha) fechas\_1 = pd.date\_range(start= fecha, end= pd.to\_datetime('10/01/2020',dayfirst=True)) print(fechas\_1) # Días desde el 3 de enero de 2020 hasta el 10 de enero de 2020 fechas\_2 = pd.date\_range(start= fecha, periods= 8) # Equivalente al anterior. Los períodos por defecto se cuentan en días, pueden cambiarse. print(fechas\_2) fechas\_3 = pd.date\_range(start= fecha, periods= 8,freq='M') # Aquí se cuentan 8 períodos en meses, marcando como día el último de cada período. print(fechas\_3) mes\_inicio = fecha.strftime('%Y-%m') print(mes\_inicio) fechas\_4 = pd.period\_range(start= mes\_inicio,periods= 8,freq='M') # Generar 8 meses consecutivos a partir de mes\_inicio print(fechas\_4) cuanto\_tiempo = fechas\_3[7] - fechas\_3[0] # ¿Cuánto tiempo pasó desde el primer período hasta el último? Esto es un objeto Timedelta print(cuanto\_tiempo) cuanto\_tiempo\_meses = fechas\_3[7].to\_period('M') - fechas\_3[0].to\_period('M') # Lo mismo pero en meses. Hay que pasar cada término a meses print(cuanto\_tiempo\_meses) |
| --- |

Estos ejemplos muestran cómo crear y operar con índices de tiempo. Construyamos ahora series de tiempo a partir de estos índices:

| fechas\_presidentes\_orig = Presidentes\_archivo['start'] # Extraigo fechas de inicio de período presidencial print(fechas\_presidentes\_orig) print(type(fechas\_presidentes\_orig)) # Estas fechas son objetos Series. Necesito transformarlos en índices fechas\_presidentes = pd.DatetimeIndex(fechas\_presidentes\_orig) # Transformo Series en índices print(fechas\_presidentes) Serie\_presidentes = pd.Series(Presidentes\_nombres.values,index=fechas\_presidentes) # Creo un objeto Series con los nombres y las fechas de los presidentes print(Serie\_presidentes) Serie\_presidentes['1850':'1901'] # ¿Qué presidentes gobernaron entre 1850 y 1901 inclusive? Muy útil |
| --- |

De esta manera nos introducimos en el uso de las Series de Tiempo con Pandas.