PED - Tema 2: Manipulación y agrupación de datos

Creación de estructuras de datos y acceso y modificación de datos (Numpy)

- Análisis de datos con NumPy
- Arrays y computación vectorizada con NumPy
- Manipulaciń avanzada de arrays
- Broadcasting

Análisis de datos con NumPy

Bibliografía:

- Capítulo 1, apartado 1.3, sección Numpy Data Analysis (2nd Edition)
- Capítulo 1, apartado 1.3, sección Numpy Data Analysis (3rd Edition)

Arrays y computación vectorizada con NumPy

Transponse:

Se emplea el atributo `.T`.

Alternativamente se puede usar la función `.transpose`. Cuando tiene una tupla o array de argumento se altera el orden de las dimensiones.

```
a = np.arange(12).reshape((2, 2, 3))
print(a.transpose((2, 1, 0)).size())
```

```
(3, 2, 2)
```

Preguntado en: EX.20220210.T.1

Splits en arrays de NumPy.

Preguntado en: EX.20250206.T.2

Universal functions (ufuncs).

El atributo `nparray.values` de un array NumPy devuelve un array estándar de Python.

Preguntado en: EX.20250904.T.1

nparray.add.reduce():

`np.add.reduce()` es equivalente a `.sum()`

No parece que se haya preguntado nunca. Aparece en el Apéndice A.4 “Advance ufunc Usage” de DAfP 3.

nparray.add.accumulative():

`accumulative()` es a `reduce()` lo que `cumsum()` es a `sum()`.

No parece que se haya preguntado nunca. Aparece en el Apéndice A.4 "Advance ufunc Usage" de DAfP 3.

np.where():

`np.where()` se presenta en el apartado "Combining Data with Overlap" del capítulo 8 y no ha aparecido en ningún exámen, pero puede ser útil.

Dependiendo de la matriz booleana del primer argumento pone el valor del parámetro 2 si es verdadero y el parámetro 3 si es falso.

```
df["columna1"] = np.where(df["stock"] < 5, "Sí", "No")
```

Bibliografía:

- Capítulo 4, introducción y apartados 4.1, 4.2, 4.3, 4.5 y 4.6 (2nd Edition)
- Capítulo 4, introducción y apartados 4.1, 4.3, 4.4, 4.6 y 4.2 (3rd Edition)

Manipulación avanzada de arrays

np.select():

En Python básico, cuando hay que aplicar condicionales se usan sentencias como `if` y sus variantes. Sin embargo, en análisis de datos su uso no es práctico ya que podríamos estar hablando de decenas de condiciones encadenadas.

NumPy provee la función `np.select()` para condicionales, donde:

```
notas = pd.Series([3, 5, 7, 9])
condiciones = [notas < 5, notas < 7, notas < 9]
valores = ["suspenso", "aprobado", "notable"]
resultado = np.select(condiciones, valores, default="sobresaliente")
print(resultado)
```

Bibliografía:

- Apéndice A, apartado A2, sección Reshaping Arrays y apartado A3 (2nd Edition)
- Apéndice A, apartado A2, sección Reshaping Arrays y apartado A3 (3rd Edition)

Broadcasting

Para obtener un vector de máximos por broadcasting se puede usar `.transform("max")`, que aplica el valor correspondiente a un grupo a todos los valores del grupo.

```
max_por_fila = df.groupby("pais")["ventas"].transform("max")
```

Bibliografía:

- Apéndice A, apartado A3 (2nd Edition)

- Apéndice A, apartado A3 (3rd Edition)

Importación y exportación de datos (Numpy)

Bibliografía:

- Básica
  - Capítulo 4, apartado 4.4 Python for Data Analysis (2nd Edition)
  - Capítulo 4, apartado 4.5 Python for Data Analysis (3rd Edition)

Creación de estructuras de datos (Pandas)

Puedes consultar la Referencia de la API de Pandas.

Bibliografía:

- Básica
  - Capítulo 5, introducción y apartado 5.1 Python for Data Analysis (2nd Edition)
  - Capítulo 5, introducción y apartado 5.1 Python for Data Analysis (3rd Edition)

Acceso y modificación de datos (Pandas)

`.unique()` aplicado a una serie devuelve los valores únicos encontrados.

df.sum():

La función `df.sum()` suma los valores por filas o columnas.

El sumarse filas o columnas se marca con el atributo `axis`, que indica la dirección en la que se suman las agrupaciones. Está por defecto con el valor `x` o `rows` por lo que suma a lo largo de la dirección de las filas, es decir, cada columna.

Para sumar a lo largo de las columnas (cada fila), se debe marcar el atributo `axis=columns`.

Preguntado en: EX.20250206.T.1.

`sum(skipna=False)` indica que si hay al menos un `NaN` en la fila, el resultado de la suma será `NaN`.

Preguntado en: EX.20250206.T.1.

Bibliografía:

- Básica
  - Capítulo 5, apartados 5.2 y 5.3 Python for Data Analysis (2nd Edition)
  - Capítulo 5, apartados 5.2 () y 5.3 (y) Python for Data Analysis (3rd Edition)

Agregación de datos y operaciones por grupo

La función `DataFrame.groupBy()` sirve para hacer agrupaciones de valores en función de la/s columna/s que se le indica como primer argumento.

Por defecto los agrupadores se convierten en índices, salvo que se ponga `index_of=False`.

```
pd.groupby("column", as_index=False)
```

El argumento `observed` aplica a las variables categóricas, e indica si se debe excluir o no aquellas categorías que tienen valor 0, es decir, si mantiene solos las categorías "observadas".

```
df["categoria"] = df["categoria"].astype("category")
df.groupby("categoria", observed=True)["compras"].sum()
```

Para concatenar DataFrame's se usa la función `concat()`, que suele ir acompañada del parámetro `ignore_index=True` para que no mantenga los índices originales de los `DataFrame`'s.

```
df = pd.concat([df, dfc], ignore_index=True)
```

count():

`df.count()` cuenta el número de filas, se salta al contar los `NaN`, y devuelve un `DataFrame`.

Por otra parte, `df.size()` cuenta el número de filas, cuenta también los `NaN` y devuelve una `Series`.

`df.count_values()` cuenta el número de veces.

Bibliografía:

- Capítulo 10 Python for Data Analysis (2nd Edition)
- Capítulo 10 Python for Data Analysis (3rd Edition)

Limpieza y preparación de datos

pd.Categorical:

Para definir una categoría ordenada:

```
df["columna"] = pd.Categorical(df["columna"], categories=orden_meses, ordered=True)
```

s.loc[]:

La función `s.loc[]` selecciona elementos en una `Series` usando como parámetro los nombres de los índices de la serie que debe devolver.

Cuando se devuelve un valor negativo o incorrecto, devuelve un error.

Toma nota de que `loc` va acompañado de corchetes, no de paréntesis.

s.take():

La función `s.take()` selecciona elementos de una `Series` usando como parámetro un array con las posiciones de índices que debe devolver.

Cuando `s.take()` tiene un índice negativo cuenta desde atrás.

```
sub_s = s.take([0, 2, -1])
```

Se preguntó en: EX.20250904.T.4 (se explica en el apartado 7.2 de la 3.ª, que no está en principio en temario, y no se menciona en temario los índices negativos)

s.reindex():

`s.reindex()` reordena las columnas en una serie (no selecciona como `loc[]` o `take()`) en base a los nombres de índices que se pasan como argumentos.

Si se introduce un valor negativo o incorrecto, pone el valor `NaN` en esa posición.

set_index:

Convierte las columnas en índice.

Las columnas que se convierten en índice se borran, salvo que añadas el argumento `drop=False`.

```
df2 = df.set_index("id")
```

reset_index:

`reset_index()` hace lo contrario que `set_index`, moviendo los índices a las columnas.

Bibliografía:

- Básica
  - Capítulo 12, apartado 12.1 (2nd Edition)
  - Capítulo 7, apartado 7.5 (3rd Edition)

Importación y exportación de datos (Pandas)

pd.read_csv():

Para importar un fichero CSV directamente como DataFrame de Pandas se ejecuta la función `pd.read_csv()`.

El parámetro `names` indica los nombres de las columnas, en caso de que el CSV carezca de una fila de cabecera.

Ejemplo de código para importar varios ficheros donde la primera línea contiene un valor individual y el resto son datos sin cabecera:

```
nombres_columnas=["columna1", "columna2", "columna_linea1"]
resto_columnas = nombres_columnas[:-1]
columna_linea1 = nombres_columnas[-1:]

df = pd.DataFrame()
```

```
for path in paths:
    with open(path, "r") as f:
        # Lee la primera línea eliminando el salto de línea
        primeralinea = f.readline().strip()
        # Lee el resto de líneas, indicando la cabecera explícitamente
        dfc = pd.read_csv(f, names=resto_columnas)
        # Añade el contenido de primera línea como nueva columna
        dfc[columna_linea1] = primeralinea
        # Concatena el df final con el df del bucle
        df = pd.concat([df, dfc], ignore_index=True)
```

Se ha pedido esto en varios exámenes: EX.20250206.P.2.

Bibliografía:

- Básica
  - Python for Data Analysis. 2nd ed. Capítulo 6
  - Python for Data Analysis. 3rd ed. Capítulo 6