PythonBasicCourse-es009

October 4, 2022

1 Curso básico de Python

1.1 Apuntes

Curso básico de Python. Apuntes por Marcelo Horacio Fortino. Versión 2.2. Agosto 2022.

Versión	Autor/es	Fecha	Observaciones
1.0	Marcelo Horacio Fortino	2021/Marzo	Curso Python
1.1	Marcelo Horacio Fortino	2021/Junio	Convertido a markdown - ipynb
1.2	Marcelo Horacio Fortino	2021/Agosto	Actualizados contenidos
1.3	Marcelo Horacio Fortino	2021/Octubre	Agregado Flask microframework
1.4	Marcelo Horacio Fortino	2021/Noviembre	Agregado Pandas - datascience
1.5	Marcelo Horacio Fortino	2021/Diciembre	Agregado Devops - Ansible
2.0	Marcelo Horacio Fortino	2022/Abril	Nueva estructura: core / module
2.1	Marcelo Horacio Fortino	2022/Junio	Módulo apuntes intermedio
2.2	Marcelo Horacio Fortino	2022/Agosto	Actualizado temario y ejercicios

Estos apuntes se basan en: - La documentación oficial de Python, https://docs.python.org/es/3/tutorial/index.html, - La bibliografía presentada al final de este documento, y - Documentación propia recogida a lo largo de los años de diversas fuentes.

1.2 Objetivo del curso

- Objetivo general del curso:

- Aprender a usar Python para crear scripts y programas simples plenamente funcionales, desarrollar aplicaciones web y realizar análisis de datos.

- Objetivos específicos:

- Reconocer las características principales de Python y su utilidad práctica.
- Identificar tipos de datos (simples y compuestos) y operadores.
- Aplicar variables y estructuras de control de flujo.
- Construir funciones y clases (POO).
- Clasificar los módulos y paquetes por sus funcionalidades y objetivos.
- Comparar el lenguaje con otros similares de scripts, procedimentales y orientados a objetos.
- Probar bibliotecas para conexiones REST a aplicaciones web y bases de datos.
- Resolver problemas y errores en el código fuente proponiendo soluciones alternativas (refactoring).
- Utilizar la biblioteca pandas junto con matplotlib y numpy para realizar análisis estadísticos de datos y gráficos.

- Como resultado práctico al final del curso cada estudiante habrá creado una aplicación web utilizando el microframework de Python Flask.

1.3 Temario

- Introducción, instalación y compilación
- Datos, expresiones y sentencias
- Variables y funciones, control de flujo
- Clases y objetos, herencia, polimorfismo
- Entradas y salidas con Python
- Gestión de módulos, paquetes y bibliotecas
- Servicios y programas en red, REST API
- Desarrollo de aplicaciones web con Flask
- Análisis de datos con pandas, matplotlib y numpy

- Módulos opcionales
  - SQL ejemplos con pandas
  - Plotting con Python
  - Machine Learning con Python
  - DevOps con Ansible
  - SDK de GCP para Python
  - Kubernetes con Python y Docker

1.4 Bibliografía

- Downey, A., Elkner, J., Meyers, C. Aprenda a Pensar Como un Programador con Python. (2015).
  Recuperado de https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf
- Kent D. Lee. Python, Programming Fundamentals Second Edition. 2014.

- Marzal Varó, A., Gracia Luengo, I., García Sevilla, Pedro. Introducción a la programación con Python 3. (2014).

Recuperado de http://repositori.uji.es/xmlui/handle/10234/102653

- Miller, B., Ranum, D. Solución de problemas con algoritmos y estructuras de datos usando Python. Traducido por Mauricio Orozco-Alzate, Universidad Nacional de Colombia - Sede Manizales.
  Recuperado de https://runestone.academy/ns/books/published/pythoned/index.html?mode=browsing

- Shaw, Z. A., Learn Python 3 the Hard Way. (2016).
  Recuperado de https://learnpythonthehardway.org/

- Van Rossum, G. and the Python development team. Documentación de Python en español. (2020).
  Recuperado de https://python-docs-es.readthedocs.io/es/3.10/

2 Análisis de datos con python

- Introducción, La biblioteca pandas, Accediendo al DataFrame, Indexado en pandas, Filtros, Ordenar datos, Actualización de datos, Métodos para cambiar datos, Agregar y/o eliminar columnas y filas, Gráficos en pandas, Agrupar datos, Agregar datos, Exportar datos.

2.1 Introducción

- A partir de la versión 3.4 python cuenta con un módulo para calcular estadísticas matemáticas de datos numéricos (de tipo Real).
- Es un módulo con funciones básicas a nivel de calculadora científica, por lo tanto no reemplaza a las bibliotecas NumPy o SciPy, ni al software propietario profesional como Minitab, SAS o Matlab.
- Entre las funciones estadísticas disponibles, se pueden mencionar las medias aritmética, geométrica, harmónica, y la mediana; entre otras.

- Fuente: https://docs.python.org/es/3/library/statistics.html.

- Algunos paquetes útiles para análisis de datos:
  - Pandas, Numpy, Scipy (estadísticas)
  - Matplotlib, seaborn (visualización de datos, gráficos)
  - Scikit-Learn (Machine Learning)

3 La biblioteca pandas

- Los ejemplos desarrollados en estos apuntes han tomado como guía a la serie *Python Pandas Tutorial* de Corey Schafer
- https://www.youtube.com/watch?v=ZyhVh-qRZPA&list=PL-osiE80TeTsWmV9i9c58mdDCSskIFdDS

- Para realizar esta práctica se deberá tener python y el módulo jupyter instalado en la máquina local.
- Otro opción es abrir una cuenta gratuita en Colaboratory:
- https://colab.research.google.com/

- *Pandas es una herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python.*
- Fuente: https://pandas.pydata.org/.
- Bibliografía: McKinney, Wes and the Pandas Development Team. pandas: powerful Python data analysis toolkit. Release 1.4.2. (2022).
- Recuperado de https://pandas.pydata.org/docs/pandas.pdf.

- Documentación recomendada para comenzar a trabajar con pandas:

- La guía oficial: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
- 10 minutes to Pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
- Cookbook: https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html
- Pandas Cheat Sheet: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

[ ]:
```
# Obtener de Internet y descomprimir ficheros utilizando Python
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile
```

[3]:
```
# Obtener el fichero con los datos de https://insights.stackoverflow.com/survey
# Pertenecen a la encuesta anual de desarrolladores año 2021
zip_url = 'https://info.stackoverflowsolutions.com/rs/719-EMH-566/images/
 ↪stack-overflow-developer-survey-2021.zip'
```

[ ]:
```
# Mostrar el directorio actual
pwd
```

[ ]:
```
# Extraer el fichero zip en el directorio actual
with urlopen(zip_url) as zipresp:
    with ZipFile(BytesIO(zipresp.read())) as zfile:
        zfile.extractall('/home/usuario')
```

[ ]:
```
pip install --upgrade pip
```

[ ]:
```
pip install pandas
```

[4]:
```
import pandas as pd
```

[ ]:
```
pwd
```

- Los dos componentes principales de pandas son las *Series* y el *DataFrame*.
- Una *Serie* es básicamente una columna.
- Un *DataFrame* es una tabla multi-dimensional compuesta de *Series*.

[5]:
```
# Se crea el dataframe
datos = pd.read_csv('survey_results_public.csv')
```

[ ]:
```
datos
```

[4]:
```
# shape muestra la cantidad de filas y columnas del fichero en forma de tupla
datos.shape
```

[4]:
```
(83439, 48)
```

[ ]:
```
# Muestra la información sobre el dataframe
datos.info()
```

[6]:
```
# Muestra todas las columnas del fichero
pd.set_option('display.max_columns', 48)
pd.set_option('display.max_rows', 50)
```

[6]:
```
# Se crea el schema del dataframe
schema_datos = pd.read_csv('survey_results_schema.csv')
```

[ ]:
```
schema_datos
```

[ ]:
```
# Muestra los 5 primeros registros del dataframe
datos.head()
```

[ ]:
```
# Muestra los últimos 10 registros
datos.tail(10)
```

3.1 Accediendo al DataFrame

- Los *DataFrames* son matrices (*two dimensional arrays*) de series de dos dimensiones (filas y columnas).
- Las *Series* (columnas) son listas de filas de datos (*one dimensional arrays*).

[11]:
```
# Accede a los datos de la columna 'Employment'
datos.Employment
```

[11]:
```
0        Independent contractor, freelancer, or self-em…
1                                     Student, full-time
2                                     Student, full-time
3                                     Employed full-time
4        Independent contractor, freelancer, or self-em…
                              …
83434                                 Employed full-time
83435    Independent contractor, freelancer, or self-em…
83436                                 Employed full-time
83437                                 Employed full-time
83438                                 Employed full-time
Name: Employment, Length: 83439, dtype: object
```

[ ]:
```
# Accede a los datos de las columnas 'Employment', 'Country', 'YearsCode'
datos[['Employment', 'Country', 'YearsCode']]
```

[13]:
```
# Muestra las columnas del dataframe
datos.columns
```

[13]:
```
Index(['ResponseId', 'MainBranch', 'Employment', 'Country', 'US_State',
       'UK_Country', 'EdLevel', 'Age1stCode', 'LearnCode', 'YearsCode',
       'YearsCodePro', 'DevType', 'OrgSize', 'Currency', 'CompTotal',
       'CompFreq', 'LanguageHaveWorkedWith', 'LanguageWantToWorkWith',
       'DatabaseHaveWorkedWith', 'DatabaseWantToWorkWith',
       'PlatformHaveWorkedWith', 'PlatformWantToWorkWith',
       'WebframeHaveWorkedWith', 'WebframeWantToWorkWith',
       'MiscTechHaveWorkedWith', 'MiscTechWantToWorkWith',
       'ToolsTechHaveWorkedWith', 'ToolsTechWantToWorkWith',
       'NEWCollabToolsHaveWorkedWith', 'NEWCollabToolsWantToWorkWith', 'OpSys',
       'NEWStuck', 'NEWSOSites', 'SOVisitFreq', 'SOAccount', 'SOPartFreq',
       'SOComm', 'NEWOtherComms', 'Age', 'Gender', 'Trans', 'Sexuality',
       'Ethnicity', 'Accessibility', 'MentalHealth', 'SurveyLength',
       'SurveyEase', 'ConvertedCompYearly'],
      dtype='object')
```

- *iloc* realiza la *indexación puramente basada en la ubicación de enteros para la selección por posición.*
- Fuente: https://pandas.pydata.org/docs/reference/api/pandas.Series.iloc.html.
- Básicamente significa que accede a los elementos de la lista mediante la posición del índice.

[ ]:
```
# Accede a filas con iloc (integer location)
datos.iloc[0]
```

[ ]:
```
datos.iloc[[0, 3, 5]]
```

[20]:
```
# Muestra la columna 3 de las filas 0, 3, y 5
datos.iloc[[0, 3, 5], 3]
```

[20]:
```
0                    Slovakia
3                     Austria
5    United States of America
Name: Country, dtype: object
```

- *loc* en cambio *accede a un grupo de filas y columnas por etiqueta(s) o una matriz booleana.*
- Fuente: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html.

[ ]:
```
# loc muestra resultados buscando por etiquetas (labels)
datos.loc[[0, 3, 5], 'Country']
```

[22]:
```
# Cuenta los países de donde provienen los programadores
datos.Country.value_counts()
```

[22]:
```
United States of America                                    15288
India                                                       10511
Germany                                                      5625
United Kingdom of Great Britain and Northern Ireland         4475
Canada                                                       3012
                                                             …
Saint Kitts and Nevis                                           1
Dominica                                                        1
Saint Vincent and the Grenadines                                1
Tuvalu                                                          1
Papua New Guinea                                                1
Name: Country, Length: 181, dtype: int64
```

3.2 Indexado en pandas

[ ]:
```
# Determinar una columna como identificador primario
datos.set_index('ResponseId')
# Para establecerla
# datos.set_index('ResponseId', inplace=True)
# Para volver al ID original
# datos.reset_index(inplace=True)
```

[ ]:
```
datos
```

[30]:
```
# Determinar una columna como identificador primario al crear el dataframe
schema_datos = pd.read_csv('survey_results_schema.csv', index_col='qname')
```

[ ]:
```
schema_datos
```

[32]:
```
# Se accede al dato a través del campo 'Country'
schema_datos.loc['Country', 'question']
```

[32]:
```
'Where do you live? <span style="font-weight: bolder;">*</span>'
```

[ ]:
```
# Ordenar los datos: ascendente
schema_datos.sort_index()
# Ordenar los datos: descendente
# schema_datos.sort_index(ascending=False)
```

3.3 Filtros

[ ]:
```
# Muestra todos los registros que cumplen con el criterio
filtro = datos['Country'] == 'Spain'
```

[ ]:
```
datos[filtro]
```

[42]:
```
# Datos filtrados usando loc
datos.loc[filtro, 'Employment']
```

[42]:
```
11                                        Employed full-time
68                                        Employed full-time
79                                        Employed full-time
108                                       Employed full-time
267                                       Employed full-time
                                  …
83097                                     Student, part-time
83133                                     Employed full-time
83137                   Not employed, but looking for work
83159                                     Student, full-time
83313    Independent contractor, freelancer, or self-em…
Name: Employment, Length: 1485, dtype: object
```

[46]:
```
prog_python = (datos['Country'] == 'Spain') & (datos['LanguageHaveWorkedWith']␣
↪== 'Python')
```

[ ]:
```
datos.loc[prog_python]
```

[48]:
```
filtrar_paises = (datos['Country'] == 'Spain') | (datos['Country'] == 'France')
```

[ ]:
```
datos.loc[filtrar_paises]
# Negación
# datos.loc[~filtrar_paises]
```

[50]:
```
salario_alto = datos['ConvertedCompYearly'] > 80000
```

[ ]:
```
datos.loc[salario_alto]
```

[ ]:
```
datos.loc[salario_alto, ['Country', 'LanguageHaveWorkedWith']].head(10)
```

[53]:
```
salario_eur = (datos['ConvertedCompYearly'] > 80000) & (datos['Currency'] ==␣
↪'EUR European Euro')
```

[ ]:
```
datos.loc[salario_eur]
```

[55]:
```
datos['LanguageHaveWorkedWith']
```

[55]:
```
0        C++;HTML/CSS;JavaScript;Objective-C;PHP;Swift
1                                  JavaScript;Python
2                            Assembly;C;Python;R;Rust
3                               JavaScript;TypeScript
4                     Bash/Shell;HTML/CSS;Python;SQL
                                …
83434                            Clojure;Kotlin;SQL
```

```
83435                                                NaN
83436                                  Groovy;Java;Python
83437                Bash/Shell;JavaScript;Node.js;Python
83438             Delphi;Elixir;HTML/CSS;Java;JavaScript
Name: LanguageHaveWorkedWith, Length: 83439, dtype: object
```

- Verifica si un patrón está contenido en una cadena. Se puede utilizar para NaN.
- Fuente https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.contains.html

[56]:
```
filtrar_lenguajes = datos['LanguageHaveWorkedWith'].str.contains('Python',␣
↪na=False)
```

[ ]:
```
filtrar_lenguajes
```

[ ]:
```
datos.loc[filtrar_lenguajes, 'LanguageHaveWorkedWith']
```

3.4 Ordenar datos

[ ]:
```
datos.sort_values(by="Country")
```

[ ]:
```
datos.sort_values(by="Country", ascending=False)
```

[ ]:
```
datos.sort_values(by=["Country", "ConvertedCompYearly"])
```

[ ]:
```
# COUNTRY ascendente, ConvertedCompYearly descendente
datos.sort_values(by=["Country", "ConvertedCompYearly"], ascending=[True,␣
↪False])
```

[ ]:
```
# Modifica datos
# datos.sort_values(by=["Country", "ConvertedCompYearly"], ascending=[True,␣
↪False], inplace=True)
# Reestablece el orden original
# datos.sort_index()
```

[ ]:
```
# Ordena una columna
datos["Country"].sort_values()
```

[ ]:
```
datos["Country"].head(20)
```

[ ]:
```
datos.sort_values(by=["ConvertedCompYearly", "Country"], ascending=[False,␣
↪True]).head(50)
```

[ ]:
```
datos["ConvertedCompYearly"].nlargest(25)
```

[ ]:
```
datos.nlargest(25, "ConvertedCompYearly")
```

[ ]:
```
datos.nsmallest(25, "ConvertedCompYearly")
```

3.5 Actualización de datos

[ ]:
```
datos.columns
```

[ ]:
```
# Cambiar a mayúsculas el campo country
datos['Country'] = datos['Country'].str.upper()
```

[59]:
```
# Cambia a mayúsculas las etiquetas de los campos
datos.columns = (x.upper() for x in datos.columns)
```

[ ]:
```
datos
```

[ ]:
```
# Reemplazar caracteres en las etiquetas de los campos
datos.columns = datos.columns.str.replace('_', '')
datos
```

[ ]:
```
# Cambiar el nombre a un campo
datos.rename(columns={'RESPONSEID':'ID'})
# Para establecerlo
# datos.rename(columns={'RESPONSEID':'ID'}, inplace=True)
```

[ ]:
```
datos.loc[1]
```

[64]:
```
# Cambiar un dato a un registro
datos.loc[1, ['COUNTRY', 'LANGUAGEWANTTOWORKWITH']] = ['Spain', 'Python']
```

[ ]:
```
datos.loc[1]
```

[66]:
```
datos.loc[1, 'COUNTRY'] = 'Netherlands'
datos.at[1, 'COUNTRY'] = 'Netherlands' # Idéntico resultado a datos.loc
```

[67]:
```
filtro_modificar_pais = (datos['COUNTRY'] == 'Spain')
datos.loc[filtro_modificar_pais, 'COUNTRY'] = 'España'
```

[ ]:
```
datos[filtrar_paises]
```

[ ]:
```
datos
```

3.6 Métodos para cambiar datos

- Existen 4 métodos para cambiar datos
  - apply: llama a una función con un dataframe o serie de objetos
  - map: es más rápido que *replace*
  - applymap: solo funciona en dataframes
  - replace

[71]:
```
# Ejemplos de apply en series de objetos
# Accede a un campo y devuelve las columnas y filas
datos['COUNTRY'].apply(len)
```

[71]:
```
0         8
1        11
2        18
3         7
4        52
         ..
83434    24
83435     5
83436    24
83437     6
83438     6
Name: COUNTRY, Length: 83439, dtype: int64
```

[9]:
```
def actualizar_pais(COUNTRY):
    return COUNTRY.lower()
```

[73]:
```
datos['COUNTRY'].apply(actualizar_pais)
```

[73]:
```
0                                                 slovakia
1                                              netherlands
2                                       russian federation
3                                                  austria
4        united kingdom of great britain and northern i…
                                  …
83434                             united states of america
83435                                                benin
83436                             united states of america
83437                                               canada
83438                                               brazil
Name: COUNTRY, Length: 83439, dtype: object
```

[ ]:
```
datos['COUNTRY'] = datos['COUNTRY'].apply(lambda z: z.upper())
```

[ ]:
```
datos
```

[ ]:
```
# Ejemplos de apply en dataframes
# Ejecuta el código en cada serie del dataframe
datos.apply(len)
```

[78]:
```
# Ejecuta el código en cada serie del dataframe
# modificando el eje a columna
datos.apply(len, axis='columns')
```

[78]:
```
0        48
1        48
2        48
3        48
4        48
         ..
83434    48
83435    48
83436    48
83437    48
83438    48
Length: 83439, dtype: int64
```

[ ]:
```
# Ejemplos de apply en series
# Ejecuta el código en cada cada valor de la serie
datos.apply(pd.Series)
```

[ ]:
```
# Ejemplos de applymap en dataframes
# Ejecuta el código (función) en cada serie del dataframe
datos.applymap
```

[81]:
```
# El método map solamente funciona con series
# Permite cambiar los valores de las series
datos['SOACCOUNT']
```

[81]:
```
0        Yes
1        Yes
2        Yes
3        Yes
4        Yes
        …
83434     No
83435    Yes
83436    Yes
83437    Yes
83438    Yes
Name: SOACCOUNT, Length: 83439, dtype: object
```

[82]:
```
datos['SOACCOUNT'].map({'Yes': True, 'No': False})
```

[82]:
```
0         True
1         True
2         True
3         True
4         True
         …
83434    False
```

```
83435     True
83436     True
83437     True
83438     True
Name: SOACCOUNT, Length: 83439, dtype: object
```

[ ]:
```
# Aplicar los cambios
datos['SOACCOUNT'] = datos['SOACCOUNT'].map({'Yes': True, 'No': False})
```

[ ]:
```
# El método replace permite cambiar los valores de las series
# Se aplican los cambios creando una variable que utilize
# este código como en el ejemplo anterior
datos['COUNTRY'].replace({'NETHERLANDS':'PAISES BAJOS'})
```

3.7 Agregar y/o eliminar columnas y filas

[ ]:
```
# Mostrar celdas con NaN
datos.isna()
```

[ ]:
```
# Obtener el porcentaje de celdas con NaN
datos.isna().mean()
```

[ ]:
```
datos.isna().mean() < .7
```

[ ]:
```
# Eliminar columnas
datos.drop(['USSTATE', 'UKCOUNTRY'], axis = 1)
```

[84]:
```
# Haciendo un join de dos columnas con un espacio para delimitarlas
datos['NEWSTUCK'] + ' ' + datos['NEWSOSITES']
```

[84]:
```
0        Call a coworker or friend;Visit Stack Overflow…
1            Visit Stack Overflow;Google it Stack Overflow
2        Visit Stack Overflow;Google it;Watch help / tu…
3        Call a coworker or friend;Visit Stack Overflow…
4        Visit Stack Overflow;Go for a walk or other ph…
                              …
83434    Call a coworker or friend;Google it Stack Over…
83435    Call a coworker or friend;Visit Stack Overflow…
83436    Call a coworker or friend;Visit Stack Overflow…
83437    Call a coworker or friend;Visit Stack Overflow…
83438    Call a coworker or friend;Visit Stack Overflow…
Length: 83439, dtype: object
```

[85]:
```
# Crea la nueva columna AYUDA
datos['AYUDA'] = datos['NEWSTUCK'] + ' ' + datos['NEWSOSITES']
```

[ ]:
```
# Eliminar columnas
# Otra opción: del datos['nombre_columna']
# Aplicar los cambios con: inplace=True
datos.drop(columns=['NEWSTUCK', 'NEWSOSITES'])
datos.columns
```

[87]:
```
# Dividir una columna
# datos[['datos1', 'datos2']] = datos['nombre_columna'].str.split(' ',␣
↪expand=True)
```

[ ]:
```
# Agregar una fila o registro
datos.append({'LANGUAGEWANTTOWORKWITH':'Python'}, ignore_index=True)
```

[91]:
```
# Agregar otro dataframe
datos_doble = datos.append(datos)
```

[ ]:
```
datos_doble
```

[94]:
```
datos_doble = datos_doble.drop_duplicates()
```

[95]:
```
datos_doble.shape
```

[95]:
```
(83439, 49)
```

[ ]:
```
# Eliminar una fila o registro
datos.drop(index=83438)
```

3.8 Gráficos en pandas

- Los métodos *plot.* se pueden aplicar a las *Series* y a los *DataFrames*.
- De forma predeterminada cada columna se grafica como un elemento distinto (area, bar, etc.)
- Cada gráfico creado con pandas es de tipo objeto *Matplotlib*.
- Para crear gráficos es necesario importar las bibliotecas *numpy*, *pandas* y *matplotlib*.

[3]:
```
import numpy as np
import pandas as pd
```

- *Matplotlib es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python.*
- Fuente: https://matplotlib.org/.

[ ]:
```
pip install matplotlib
```

[5]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
# Diagrama de caja de todas las columnas con datos numéricos
boxplot = datos.boxplot(figsize = (5,5))
```

[ ]:
```
# Diagrama de caja de una columna
datos.boxplot(column =['CONVERTEDCOMPYEARLY'], grid = False)
```

[ ]:
```
# Elimina todos los valores superiores a 300000 U$S anuales para el salario
datos.drop(datos[datos['CONVERTEDCOMPYEARLY'] > 300000.0].index)
```

[ ]:
```
# Elimina todos los valores inferiores a 100 U$S anuales para el salario
datos.drop(datos[datos['CONVERTEDCOMPYEARLY'] < 1000].index, inplace = True)
```

[ ]:
```
datos.drop(datos[datos['CONVERTEDCOMPYEARLY'] > 200000].index, inplace = True)
```

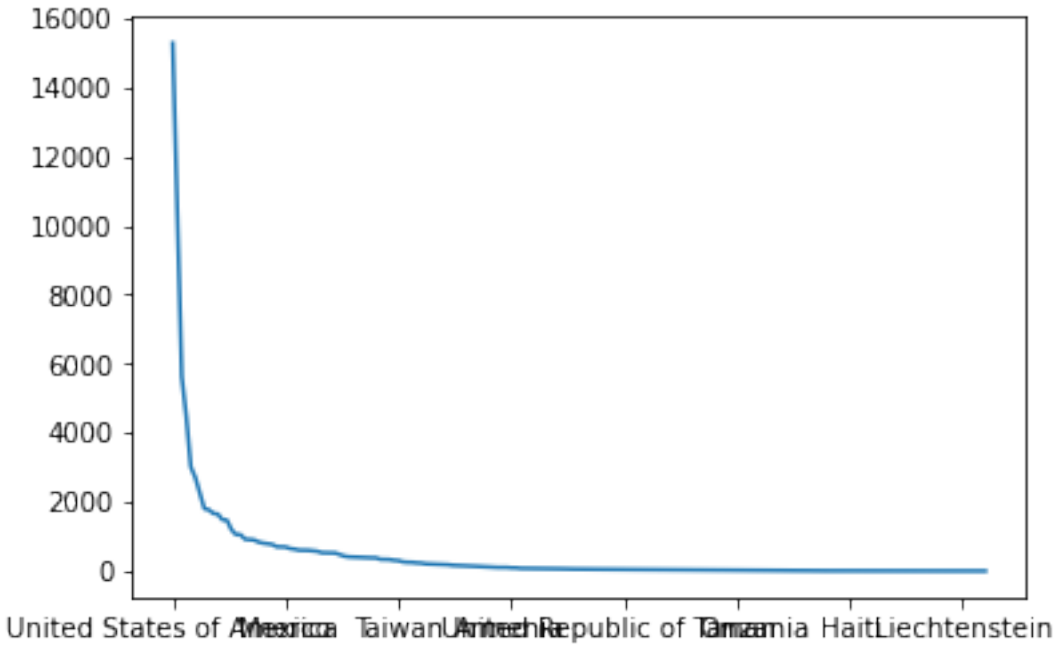
[ ]:
```
# diagrama de caja con los cambios
datos.boxplot(column =['CONVERTEDCOMPYEARLY'], grid = False)
```

[ ]:
```
# Histograma con los mismos datos
histograma = datos.hist(column='CONVERTEDCOMPYEARLY', bins=10, grid=True,
                        figsize=(12,8), color='#55bf91', zorder=2, rwidth=0.9)
```

[6]:
```
# Creamos la variable grafico
grafico = (datos.COUNTRY.value_counts())
```

[7]:
```
grafico.plot()
```

[7]: <AxesSubplot:>



[ ]:
```
datos["CONVERTEDCOMPYEARLY"].plot()
```

[ ]:
```
# Seleccionamos los primeros diez valores
grafico_cant_prog = (grafico.nlargest(10))
```

[ ]:
```
# Utilizamos título, etiquetas y colores
grafico_cant_prog.plot.bar(title='Cantidad de programadores por país',␣
↪xlabel='Paises',
                           ylabel='# de programadores', color=['red', 'green',␣
↪'blue'])
```

[ ]:
```
# Utilizamos una paleta de colores
grafico_cant_prog.plot.bar(title='Cantidad de programadores por país',
                           xlabel='Paises', ylabel='# de programadores',␣
↪colormap='plasma')
```

[ ]:
```
# Métodos disponibles en Pandas para gráficos
[
    nombre
    for nombre in dir(datos.plot)
    if not nombre.startswith("_")
]
```

[ ]:
```
datos["CONVERTEDCOMPYEARLY"].plot.box(xlabel="Salario anual", ylabel="Miles de␣
↪euros")
```

3.9 Agrupar datos

[ ]:
```
datos.describe()
```

[ ]:
```
datos["CONVERTEDCOMPYEARLY"].mean()
```

[ ]:
```
datos.median()
```

[ ]:
```
datos["MAINBRANCH"].value_counts()
```

[ ]:
```
# datos["NEWOTHERCOMMS"].value_counts()
# normalize=True convierte a porcentaje los valores
datos["NEWOTHERCOMMS"].value_counts(normalize=True)
```

3.10 Agregar datos

[ ]:
```
pais = datos.groupby(["COUNTRY"])
```

[ ]:
```
pais.get_group("ESPAÑA")
```

[ ]:
```
pais["MAINBRANCH"].value_counts().head(20)
```

[ ]:
```
pais["CONVERTEDCOMPYEARLY"].median().head(20)
```

[ ]:
```
pais["CONVERTEDCOMPYEARLY"].agg(["median", "mean", "count"])
```

3.11 Exportar datos

[ ]:
```
filtro = datos['COUNTRY'] == 'Spain'
datos_españa = datos.loc[filtro]
datos_españa.head()
```

[ ]:
```
print(datos_españa)
```

[ ]:
```
# exportar a .csv
datos_españa.to_csv('datos_españa.csv')
```

[ ]:
```
# exportar a Excel
pip install openpyxl
```

[ ]:
```
datos_españa.to_excel('datos_españa.xlsx')
```