Disclaimer: Notebooks translated to Spanish are distributed as an optional aid to assist in your learning and comprehension. We make no guarantees that the translations are completely accurate nor that the translated code blocks will run properly.

Caso previo: Configuración de Python, Jupyter Notebook y Git

Objetivos (2 min)

Al final de este caso debe haber instalado y estar familiarizado con Python, Jupyter, Anaconda y GitHub.

Además, deberá tener un conocimiento práctico de varios componentes fundamentales de Python (enumerados al final del caso). Es decir, ser capaz de escribir código simple para realizar tareas analíticas como la creación de métricas para la toma de decisiones de negocio.

Python y Jupyter (2 min)

Python es un lenguaje de programación de propósito general que permite el análisis de datos, tanto simples como complejos. Python es increíblemente versátil, permitiendo a los analistas, consultores, ingenieros y gerentes obtener y analizar información para la toma de decisiones informadas (a partir de revelaciones obtenidas de los datos).

Jupyter Notebook es una aplicación web de código abierto (open-source) que permite el desarrollo de código Python. Jupyter permite además graficación interactiva (inline) y proporciona mecanismos útiles para la visualización de datos que lo convierten en un excelente recurso para una variedad de proyectos y disciplinas.

En la siguiente sección se explicará cómo instalar y comenzar a trabajar con Python y Juypter.

Configurando el entorno Python (5 min)

A continuación se incluyen las instrucciones guía para Windows y MacOS. Siga la que corresponda a su sistema operativo.

Instalación en Windows

1. Abra su navegador y vaya a https://www.anaconda.com/distribution/
2. Haga clic en “Windows” y luego en “Descargar” (Download) para obtener el instalador de 64 bits de Python 3.7
3. Ejecute el archivo descargado que se encuentra en la sección de descargas (paso 2)
4. Haga click en las indicaciones de las ventanas de instalación
5. Vaya al menú (o al Explorador de Windows), encuentre la carpeta de Anaconda3, y haga doble click para ejecutarla.

MacOS Install

1. Abra su navegador y vaya a https://www.anaconda.com/distribution/
2. Haga click en “MacOS” y luego en “Descargar” (Download) para el instalador de 64 bits de Python 3.7
3. Ejecute el archivo descargado que se encuentra en sus descargas (paso 2)
4. Haga click en las indicaciones de las ventanas de instalación

5. Utilizando la búsqueda de Spotlight, escriba "Anaconda" seleccione y ejecute el programa Anaconda Navigator. Recuerde que macOS viene con Python preinstalado, pero esta distribución no debe utilizarse, pues es exclusiva del sistema. Anaconda ejecutará una nueva instalación de Python; debe asegurarse de utilizar esta instalación en futuras tareas.

Manejo de archivos con Python y Jupyter (1 min)

Es una práctica común tener una carpeta principal donde se ubican todos los proyectos (por ejemplo, "investigacion_jupyter"). Las siguientes son pautas que puede aplicar para proyectos Python, las cuales pueden ayudar a mantener su código organizado y accesible:

1. Crear subcarpetas para cada proyecto Jupyter
2. Agrupar en la misma carpeta los archivos `.ipynb` (formato de archivo para los 'Notebook' de Jupyter) que estén relacionados entre si

3. Crear una carpeta de "Datos" dentro de las carpetas de proyectos individuales si se utiliza un gran número de archivos de datos relacionados

¡Con esto está listo para empezar a codificar en Python!

Configurando Git, GitHub y clonando un repositorio (7 min)

Git es un sistema de control de versiones distribuido, gratuito y de código abierto (open-source). Git es muy útil, tanto para proyectos de software sencillos, como para aquellos grandes proyectos multifuncionales que abarcan miles de archivos. El sitio web principal para la documentación de Git es https://git-scm.com/.

En esta guía encontrará una breve introducción a Git para poner en marcha el software y comprender cómo funciona su sistema de control de versiones. También, usted creará un nuevo repositorio y alcanzará a clonar un repositorio de GitHub.

GitHub es una plataforma online que alberga repositorios para que los usuarios interactúen en sus propios proyectos y en proyectos de colaboración.

Aquí tiene un Tutorial de Git que puede ver en tu tiempo libre para avanzar en su comprensión de este sistema de intercambio de archivos distribuidos.

Existen numerosas Interfaces gráficas especializadas (GUI) para Git. De hecho, en los ejemplos dados puede encontrar diferentes editores para Git, de los cuales puede elegir el de su preferencia. También le recomendamos buscar un tutorial introductorio simple sobre el editor de su elección.

Los siguientes pasos describen cómo empezar a trabajar con Git, crear una cuenta en GitHub, crear un repositorio, y clonar un repositorio en su computador:

1. Abra su navegador, dirijase a https://git-scm.com/downloads y descargue Git para su sistema operativo.

2. Cree un nuevo repositorio navegando a la carpeta de su computador en la que te gustaría trabajar. Desde esta carpeta, ejecute el comando: `git init`.

3. Abra su navegador y dirijase a https://github.com/ y registre su cuenta de Github.

4. En GitHub, navegue a la página principal del repositorio que te interesa utilizar. Tenga en cuenta que cuando crea un repositorio en GitHub, es un ***repositorio remoto***. Se clonará un repositorio para crear una copia local en su computador y así poder sincronizar entre las dos ubicaciones cuando sea necesario. De esta manera, se tendrá una copia ***local*** del repositorio y una copia ***remota*** del mismo.

5. Bajo el nombre del repositorio en Github, haga click en clonar o descargar.

6. De la sección *Clonar con HTTPS*, copie completa la línea que incluye el enlace URL para clonar.

7. Abra su *terminal* de línea de comandos y navegue desde el directorio de trabajo actual hasta la ubicación donde desea que se haga el directorio clonado.

8. Escriba y ejecute el comando: `git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY` (es decir, pegue el texto que habia copiado antes en el paso 6).

Ya que usted debió haber creado su clon local del repositorio, cubramos algunas de las utilidades disponibles con Git.

Comprendiendo el flujo de trabajo de Git (8 min)

Git es un sistema de control de versiones distribuido. Esto significa que te permite rastrear los *cambios* de un conjunto de archivos. Puede que esté acostumbrado a ver archivos con nombres como *"Ensayo(2)(2)_Final(2)_edición2Mazo(3)EsteSi.docx"* mientras intenta mantener sincronizadas diferentes versiones de un archivo cuando es editado por varias personas. Git resuelve este problema rastreando los cambios como **commits** específicos, permitiendo volver atrás en el tiempo a versiones anteriores, y fusionar versiones de archivos trabajados por diferentes personas.

Git es una herramienta avanzada y altamente compleja con una curva de aprendizaje empinada, así que no se preocupe si no todo tiene sentido desde el principio. Solo necesitará aprender unos pocos conceptos para empezar a usar Git. Específicamente, necesitará saber cómo:

- Indicar qué archivos quiere rastrear usando Git
- Confirmar los cambios en estos archivos en momentos específicos
- Enviar (push) estos cambios a un host remoto

Fíjese en la diferencia entre "git" (el software que se ejecuta localmente y que se ha utilizado durante años) y "GitHub" (la plataforma web o host más popular para los repositorios de git). Usted utilizará el software `git` en su máquina local, y luego enviará su código a GitHub para compartir, colaborar y para tener una copia de seguridad de sus archivos.

Los comandos más importantes de git que tendrá que usar al principio son: * `git add` (que indica qué archivos quieres que git rastree) * `git commit` (que indica cuándo quieres crear un "punto de control" de tus archivos rastreados) * `git push` (que indica cuándo quieres enviar los datos de tu máquina local a GitHub).

Veamos un ejemplo sencillo de confirmación de un cambio en un archivo de texto "hola.txt":

1. Cree un archivo de texto `hola.txt` en su computador local, en la carpeta desde la que usará Git.

2. Edite el archivo para añadir una línea al archivo de texto; por ejemplo, añada "Hola Mundo" como primera línea. Guarde el archivo.

3. Haga un seguimiento de este archivo usando Git. Para hacerlo, utilice el comando: `git add hola.txt`. (Si tiene varios archivos y quieres añadir todos los archivos con cambios en la carpeta a rastrear por git, puede usar el comando: `git add *`)

4. Para confirmar los cambios en todos los archivos rastreados, use el comando: `git commit -m "Mensaje del commit"` donde el "Mensaje del commit" debería ser un recordatorio corto pero descriptivo de los cambios que ha hecho desde la última vez que ejecutó un 'commit' (para el primero, puede usar un mensaje como: "commit inicial", no obatante, usar mensajes lo más descriptivos posibles ayudará a depurar cualquier problema más adelante).

5. Ahora ha creado un punto de control de sus archivos (algo reversible más tarde, si fuera necesario), pero sigue existiendo sólo en su máquina local. Para Enviar (push) los cambios a su repositorio remoto, use el comando: `git push origin master`

Algunos otros comandos útiles que puede utilizar para comprobar el estado y la historia de su depósito incluyen:

- `git status`, que muestra si los archivos están sin rastrear, o rastreados pero tienen cambios y necesitan ser confirmados con un `commit`.
- `git log`, que te muestra una lista de mensajes de confirmación (commits) que han sido aplicados.

Lo anterior cubre lo básico de Git. Y aunque se necesita una considerable práctica para dominar estos conceptos, hemos cubierto las funcionalidades de clonación, adición, confirmación (Committing) y envío (Pushing), que son la columna vertebral de Git, como software de control de versiones.

Ahora estamos listos para empezar el caso.

Identificando oportunidades de expansión para vuelos de aerolíneas comerciales de lujo

Introducción del Caso (5 min)

Contexto del Negocio. Usted es un empleado de GrowthAir, una compañía aérea comercial en crecimiento. En los últimos años, GrowthAir ha ampliado los servicios de vuelos de lujo a lugares de todo el mundo. Siguiendo el excelente desempeño de su equipo en la identificación de nuevas oportunidades de negocio el año pasado, se le ha encomendado la tarea de identificar los principales países para ampliar aún más el servicio de vuelos de lujo de GrowthAir.

Problema de negocio. Su gerente le ha pedido que responda a la siguiente pregunta: "¿En qué países debería GrowthAir expandir su servicio de vuelos de lujo?"

Contexto analítico. Los datos relevantes son una serie de estimaciones de éxito (es decir, probabilidades de éxito) que sus equipos internos de investigación de mercado han elaborado. Utilizando su capacidad de realizar análisis de datos en Python, se embarcará en la tarea de resumir las estimaciones de éxito disponibles para producir una recomendación concisa a su jefe.

Ejecución. En este caso, observará métricas simples en los datos como la media, la mediana, min/máx, y finalmente hará su sugerencia basada en estas estadísticas elementales. El componente de análisis debe ser relativamente simple y cada estudiante debe ser capaz de hacer esto con lápiz y papel. Sin embargo, lo especial de este caso es que realizará estas tareas en un entorno Python para familiarizarse con las nuevas técnicas y herramientas.

Fundamentos de Python (30 min)

Python es un lenguaje de programación general interpretado y de alto nivel que se publicó por primera vez en 1991. Python permite a los usuarios manipular fácilmente los datos y almacenar los valores en lo que se conoce como objects. Todo en Python es un objeto y tiene un tipo (type).

Por ejemplo, si un usuario pretende almacenar el entero 5 en un objeto llamado `mi_entero`, esto se puede lograr escribiendo la sentencia Python, `mi_entero = 5`. Si estás acostumbrado a la sintaxis de las matemáticas, esto puede parecer confuso. En este caso `mi_entero` es una variable, muy parecida a la que puedes encontrar en álgebra, pero el signo `=` es para *asignación* no para *igualdad* (esta última se denota con `==`). Así que en este caso `mi_entero = 5` debe entenderse como más como `hacer que mi_entero sea igual a` en lugar de que `mi_entero es igual a`.

Aquí, `mi_entero` es un ejemplo de una *variable* porque puede cambiar de valor (más tarde podríamos asignarle un valor diferente) y es de tipo **Integer**, conocido en Python simplemente como `int`. A diferencia de otros lenguajes de programación, Python adivina el tipo de la variable a partir del valor que le asignamos, por lo que no necesitamos especificar el tipo de la variable explícitamente.

Las variables de Python tienen una variedad de tipos de datos. Si usted está acostumbrado a usar Microsoft Excel, esto es similar a la forma en que Excel distingue entre tipos de datos como Texto, Número, Moneda, Fecha o Científico. Aquí hay algunos tipos de datos comunes que encontrará en Python:

1. Enteros, escribe `int`: `mi_entero = 1`
2. Flotante (De punto flotante), tipo `float`: `mi_flotante = 25.5`
3. Cadena de caracteres, escribe `str`: `mi_cadena_de_caracteres = 'Hello'`

Note que en este caso los nombres `mi_entero`, `mi_flotante` y `mi_cadena_de_texto` son arbitrarios. Es de notar que es útil nombrar sus variables de tal manera que que los nombres contengan alguna información relacionada con su propósito, este código sería funcionalmente idéntico, si hubiéramos usado `x`, `xrtqp2` y `mi_variable_de_nombre_muy_largo`, respectivamente.

Aquí vemos que (1) Enteros y (2) Flotantes almacenan datos numéricos. La diferencia entre los dos es que los *Flotantes* almacenan variables decimales (fracciones, números de punto flotante), mientras que el tipo de *Entero* sólo puede almacenar variables enteras (números enteros). Por último, (3) es del tipo string. Las cadenas de caracteres se utilizan para almacenar datos textuales en Python (una cadena de uno o más caracteres). Más adelante en este caso usaremos variables *string* para almacenar los nombres de los países. A menudo se utilizan para almacenar identificadores como nombres de personas, nombres de ciudades y más.

Hay otros tipos de datos disponibles en Python; sin embargo, estos son los tres tipos fundamentales que verá en casi todos los programas Python. Ten siempre presente que **cada** objeto en Python tiene un tipo (type) y algunos de estos "tipos" pueden ser definidos por el usuario, lo cual es uno de los beneficios de Python.

El uso más simple de Python es como calculadora. Hemos visto cómo asignar valores a las variables, pero también podemos realizar cálculos sobre estas variables. Un programa Python contiene una o más líneas de código, que se ejecutan de forma descendente (con algunas excepciones que veremos más adelante). El siguiente programa Python demuestra esta funcionalidad:

```
x = 5
y = 7
z = x + y -1
print(z)
```

La función `print` (impresión) simplemente envía datos a su pantalla (¡No tiene nada que ver con una impresora de computador, y no desperdiciará su tinta o papel!). En el código anterior, empezamos por arriba y seguimos trabajando hasta abajo. Esto significa que asignamos el valor `5` a la variable `x`, luego el valor `7` a la variable `y` y finalmente la suma de los valores `x` y `y` menos `1` a la variable `z`. Asegúrate de que puedes crear este programa Python y que obtienes la salida `11` antes de continuar.

Hay mucho más en Python que lo anterior. En este caso, presentaremos los conceptos más básicos de Python. Si lo prefiere, primero puedes pasar por el tutorial oficial de Python más extenso y/o el tutorial de Python de la W3School y practicar conceptos más fundamentales antes de proceder con este caso.

Se recomienda especialmente que revise uno o ambos tutoriales antes o después de repasar este Notebook para consolidar e incrementar su comprensión de Python.

Ejercicio 0 (10 min):

0.1

- Use Python para calcular la respuesta a `1675 - 886` e imprímala
- Asigne su nombre a una variable de Python e imprímalo
- Calcule el resultado de `65.4 + 7 - 6 +` e imprímalo

Respuesta.

0.2

Ya mencionamos antes que las variables pueden cambiar de valor. Veamos cómo funciona esto, y también introducimos algunas operaciones más de Python:

[6]:
```
x = 4
print(x)
y = 2
x = y + x
print(x)
```

```
4
6
```

De nuevo, si estás acostumbrado a la sintaxis de las matemáticas, `x = y + x` podría parecer muy equivocado a primera vista. Sin embargo, simplemente significa *"descartar el valor existente de x, y asignar un nuevo valor que corresponde a la suma de y y x"*. Aquí podemos ver que el valor de `x` cambia, demostrando por qué las llamamos "variables".

También podemos usar más operadores que sólo `+` y `-`. Usamos`*` para la multiplicación,`/` para la división, y `**` para la potencia. También se aplican las reglas de orden de operaciones estándar de las matemáticas y los paréntesis `()` se pueden usar para redefinirlas:

[7]:
```
x = 2
y = 3
print (x + y ** x)
print((x+y)**x)
```

```
11
25
```

Ahora inténtelo usted mismo. Asigne los valores 3, 4, 5 a tres variables diferentes llamadas `a`, `b` y `c`, y calcule los valores de

- `c`al cuadrado
- `a` al cuadrado + `b` al cuadrado

Respuesta.

Ahora que hemos cubierto los fundamentos de Python, echemos un vistazo a los datos de la compañía propietaria de GrowthAir sobre las estimaciones de éxito de los países.

Explorar los datos de la compañía sobre las estimaciones de éxito (10 min)

Su compañía tiene algunos datos propietarios que necesita que usted analice. Estos datos consisten en estimaciones de la probabilidad de éxito de los proyectos de expansión global por país, creadas por varias de las principales agencias de análisis.

La variable `estimados_de_exito`, descrita a continuación, es un diccionario de Python. Antes, mostramos cómo las variables podían contener tipos de datos simples como un solo entero, o una cadena. Como una extensión de eso, las variables también pueden referirse a **estructuras de datos**. Estos son tipos de datos más complicados que comprenden muchas piezas individuales de datos, organizados de una manera específica.

Tal como antes, seguimos usando `=`, el **operador de asignación**, para asignar un valor a la variable.

El tipo (type) diccionario (dictionary) de Python almacena un mapeo de pares *llave-valor* que permite a los usuarios acceder rápidamente a la información de una llave en particular. Al especificar una llave, el usuario

puede obtener el valor correspondiente a la llave dada. La sintaxis de Python para diccionarios utiliza llaves rizadas {} (llaves):

```
mi_diccionario = {'llave1': Valor1, 'llave2': Valor2, 'llave3': Valor3}
```

El diccionario `estimados_de_exito` tiene llaves que son cadenas de texto (string), y valores que son del tipo lista (list). Las listas son estructuras de datos increíblemente útiles en Python que pueden almacenar cualquier número de objetos Python, y se denotan mediante el uso de corchetes `[]`.

Las listas presentes en `estimados_de_exito` contienen valores tipo flotantes (ver en código). Las listas son versátiles y pueden ser ampliadas añadiendo nuevos elementos al final de la lista (el lado más a la derecha se considera el final de la lista). Además, se puede acceder fácilmente a los elementos de la lista (es decir, a los objetos de la lista) mediante índices enteros. Es interesante que las listas también pueden almacenar otras listas (denominadas listas de listas). Esto las convierte en una poderosa herramienta para contener conjuntos de datos complejos.

Echemos un vistazo a los datos de `estimados_de_exito`. Cada estimación (cada ítem de cada lista) aquí es un número (flotante) entre 0 y 1 inclusive, que representa la probabilidad de que la expansión a ese país sea exitosa. Por ejemplo, tenemos 4 estimaciones de éxito para la expansión a Australia, que son 0,6 (bastante probable que tengamos éxito), 0,33 (bastante improbable que tengamos éxito), 0,11 (muy improbable que tengamos éxito) y 0,14 (también muy improbable que tengamos éxito). En general, Australia no parece una buena elección.

[9]:
```
# Las líneas que comienzan con el símbolo `#` son comentarios, y son ignoradas por␣
 ↪Python completamente
# Sólo existen para hacer tu código más legible para los humanos
# Es una buena práctica tener uno o más comentarios por encima de las partes no obvias␣
 ↪del código
# para explicar lo que hacen

# Datos sobre la probabilidad de éxito de la expansión según las estimaciones de los␣
 ↪países
estimados_de_exito = {
    'Australia': [0.6, 0.33, 0.11, 0.14],
    'France': [0.66, 0.78, 0.98, 0.2],
    'Italy': [0.6],
    'Brazil': [0.22, 0.22, 0.43],
    'USA': [0.2, 0.5, 0.3],
    'England': [0.45],
    'Canada': [0.25, 0.3],
    'Argentina': [0.22],
    'Greece': [0.45, 0.66, 0.75, 0.99, 0.15, 0.66],
    'Morocco': [0.29],
    'Tunisia': [0.68, 0.56],
    'Egypt': [0.99],
    'Jamaica': [0.61, 0.65, 0.71],
    'Switzerland': [0.73, 0.86, 0.84, 0.51, 0.99],
    'Germany': [0.45, 0.49, 0.36]
}
```

Arriba, hemos definido nuestro conjunto de datos, mapeando países específicos a listas de valores (estimaciones). Lo más sencillo que podemos hacer con estos datos es presentarlos en una salida de consola, de nuevo usando la función `print()` como sigue:

[10]:
```
#No se necesitan comillas cuando se imprime una variable existente.
print(estimados_de_exito)
```

```
{'Australia': [0.6, 0.33, 0.11, 0.14], 'France': [0.66, 0.78, 0.98, 0.2], 'Italy': [0.6],
'Brazil': [0.22, 0.22, 0.43], 'USA': [0.2, 0.5, 0.3], 'England': [0.45], 'Canada': [0.25,
0.3], 'Argentina': [0.22], 'Greece': [0.45, 0.66, 0.75, 0.99, 0.15, 0.66], 'Morocco':
[0.29], 'Tunisia': [0.68, 0.56], 'Egypt': [0.99], 'Jamaica': [0.61, 0.65, 0.71],
'Switzerland': [0.73, 0.86, 0.84, 0.51, 0.99], 'Germany': [0.45, 0.49, 0.36]}
```

Nos gustaría recomendar que la empresa se esfuerce en el país con la mayor estimación de éxito. ¿Pero qué significa esto cuando hay múltiples estimaciones de éxito para algunos países y sólo una para otros? Podemos usar métricas como promedios para resolver esto. Exploraremos esto a continuación.

Interactuando con diccionarios y listas (15 min)

Mirando cuidadosamente el diccionario de `estimados_de_exito`, notará que algunos países sólo tienen una estimación de éxito, mientras que otros tienen muchas. Por ejemplo, Inglaterra sólo tiene una estimación contenida en su lista [0,45], mientras que Jamaica tiene tres estimaciones contenidas en su lista [0,61, 0,65, 0,71]. Acerquémonos a Jamaica y demos un vistazo a algunas estadísticas resumidas de las estimaciones.

En Python, el tipo (type) diccionario tiene métodos incorporados (funciones, que analizaremos más adelante) para acceder a las llaves y valores del diccionario. Estos métodos son invocados escribiendo `.keys()` o `.values()` después del objeto del diccionario. Cambiaremos el tipo de retorno de la llamada a `.keys()` y `.values()` a una lista utilizando el método `list()`.

[11]:
```
# Miremos las llaves...
estimados_de_exito.keys()
```

[11]:
```
dict_keys(['Australia', 'France', 'Italy', 'Brazil', 'USA', 'England', 'Canada',
'Argentina', 'Greece', 'Morocco', 'Tunisia', 'Egypt', 'Jamaica', 'Switzerland',
'Germany'])
```

Note arriba que lo primero que ve es la frase "dict_keys" (llaves del diccionario), indicando el **tipo (type)** de los datos. Vamos a convertirlo en una lista que es un tipo (type) de datos más simple y más común. Podemos hacerlo pasando los datos en la función `list(...)`

[12]:
```
# Ahora veamos las llaves en forma de lista. Esto es más legible y utilizable para␣
↪nosotros más tarde.
list(estimados_de_exito.keys())
```

[12]:
```
['Australia',
 'France',
 'Italy',
 'Brazil',
 'USA',
 'England',
 'Canada',
 'Argentina',
 'Greece',
 'Morocco',
 'Tunisia',
 'Egypt',
 'Jamaica',
```

```
 'Switzerland',
 'Germany']
```

Note arriba que los datos son similares pero ya no comienzan con `dict_keys`.

[13]:
```
# También podemos ver los valores correspondientes (estimaciones) para cada llave de
 ↪la siguiente manera
list(estimados_de_exito.values())
```

[13]:
```
[[0.6, 0.33, 0.11, 0.14],
 [0.66, 0.78, 0.98, 0.2],
 [0.6],
 [0.22, 0.22, 0.43],
 [0.2, 0.5, 0.3],
 [0.45],
 [0.25, 0.3],
 [0.22],
 [0.45, 0.66, 0.75, 0.99, 0.15, 0.66],
 [0.29],
 [0.68, 0.56],
 [0.99],
 [0.61, 0.65, 0.71],
 [0.73, 0.86, 0.84, 0.51, 0.99],
 [0.45, 0.49, 0.36]]
```

Más adelante haremos uso del acceso a las llaves y valores de un diccionario para comparar las estimaciones de numerosos países. Por ahora, sólo recuerde que puede acceder a la lista completa de llaves o valores de un diccionario simplemente llamando a los métodos incorporados.

También nos gustaría comprobar si el nombre de un país es una de las llaves del diccionario. Python nos permite comprobar si una llave está en un diccionario mediante el uso de la palabra clave `in`. La frase `key in dictionary` devolverá un tipo booleano (boolean type) con valor `True (Verdadero)` si la llave es una de las llaves en el diccionario o `False (Falso)`, en caso contrario. Demos un vistazo a cómo funciona esto.

[14]:
```
print('Comprobando si la llave de Marruecos (morocco) está presente:')
print('Morocco' in estimados_de_exito) # imprime el VALOR de "Morocco in
 ↪estimados_de_exito"

print('Comprobación de la presencia de la llave de Japón:')
japan_boolean = 'Japan' in estimados_de_exito
print(japan_boolean)
```

```
Comprobando si la llave de Marruecos (morocco) está presente:
True
Comprobación de la presencia de la llave de Japón:
False
```

Ahora nos gustaría acceder al valor correspondiente a una llave específica del diccionario `estimados_de_exito`. Simplemente escriba el nombre del valor entre corchetes, junto al nombre del diccionario. Por ejemplo, `estimados_de_exito['Jamaica']` devolverá la lista de estimaciones de Jamaica:

[15]:
```
estimados_de_exito['Jamaica']
```

[15]:
```
[0.61, 0.65, 0.71]
```

Si desea almacenar el resultado en una variable que se utilizará más adelante, utilice el operador de asignación '=':

[16]:
```
lista_de_jamaica = estimados_de_exito['Jamaica']
```

Entonces puede ver el contenido de la lista a través del método `print()`:

You can then view the contents of the list via the `print()` method:

[17]:
```
print(lista_de_jamaica)
```

```
[0.61, 0.65, 0.71]
```

Para acceder a elementos específicos en un diccionario, usamos el **nombre de la llave**. Por ejemplo, utilizamos `estimados_de_exito['Jamaica']` para acceder a los datos sobre Jamaica. En cambio, para acceder a elementos específicos de una lista, utilizamos un **índice**, que indica si queremos acceder al primer elemento, al segundo elemento, etc. Python utiliza una **indexacion base cero**, lo que puede resultar confuso al principio. Esto significa que el primer elemento de la lista tiene el índice `0`, el segundo elemento tiene el índice `1`, y así sucesivamente.

Podemos acceder a cada uno de los tres primeros ítems de la variable `lista_de_jamaica` de la siguiente manera:

[18]:
```
# Cada impresión (print) se presentará en una línea nueva
print(lista_de_jamaica[0]) # imprime el primer elemento de la lista
print(lista_de_jamaica[1]) # imprime el segundo elemento de la lista
print(lista_de_jamaica[2]) # imprime el tercer elemento de la lista
```

```
0.61
0.65
0.71
```

Deslizar (rebanar) e indexar listas (entre otros objetos como `arrays` (arreglos) y `dataframes`) es una parte crucial y esencial de Python básico. Lo encontrará a menudo en su carrera de ciencias de los datos. En lugar de acceder a un solo elemento de la lista, podemos acceder a sublistas de la lista de la siguiente manera.

[19]:
```
# Tomando los últimos 2 elementos de lista_de_jamaica
print(lista_de_jamaica[-2:])  # (va desde el penúltimo elemento hasta el final)
print(lista_de_jamaica[1:])  # (va desde el segundo elemento hasta el final)
```

```
[0.65, 0.71]
[0.65, 0.71]
```

[20]:
```
# rebanando varias capas
print(estimados_de_exito['Jamaica'][0]) # obteniendo la primera estimación jamaicana␣
 ↪del diccionario
```

```
0.61
```

Arriba, hacemos las dos cosas en una sola línea de código. En la mayoría de los casos, Python opera de izquierda a derecha, por lo que primero obtendrá el elemento "Jamaica" del diccionario y luego el primer elemento (cero) de esa lista.

Python también ofrece una forma sencilla de determinar la longitud de una lista: el método `len()`. Esperamos que la longitud de `lista_de_jamaica` sea 3 ya que tiene tres elementos:

[21]:
```
len(lista_de_jamaica) # devuelve la longitud de la lista
```

[21]:
```
3
```

Ejercicio 1 (3 min):

Imprimir la longitud de las listas de estimación de éxito para Francia, Grecia y Marruecos.

Respuesta.

Exercise 2 (3 min):

¿Cuál de las siguientes opciones sería útil para almacenar las estimaciones de éxito de los proyectos si se dispusiera de ellas a nivel regional en lugar de a nivel de país?

a) Lista

b) Diccionario

c) Float

d) Cadena de caracteres (String)

Respuesta.

Ahora que estamos familiarizados con el uso de las listas y sabemos que las listas son estructuras de datos ordenadas mientras que los diccionarios son estructuras de datos no ordenadas, empecemos a comparar las estimaciones de éxito entre los países.

Cálculo del promedio de éxito de un país en específico (10 min)

Continuando nuestro análisis sobre Jamaica, la lista contiene tres números, [0,61, 0,65, 0,71]. Recordemos que estos números son del tipo `float` en Python, que almacena valores numéricos decimales. Una forma lógica de resumir estas estimaciones para que puedan ser comparadas entre países es usar el promedio aritmético. Utilicemos los operadores aritméticos básicos para calcular la estimación media de éxito para Jamaica, almacenando el resultado en una nueva variable `promedio_jamaica`.

[23]:
```
promedio_jamaica = (0.61 + 0.65 + 0.71) / 3
print(promedio_jamaica)
```

```
0.6566666666666666
```

Vemos que la probabilidad media de éxito estimada para Jamaica es de aproximadamente 0,657. Sin embargo, hemos producido esta estimación codificando manualmente los valores. Si hiciéramos esto para cada país, llevaría bastante tiempo. Así que nos gustaría utilizar una forma más automatizada de producir el promedio.

Para producir un promedio podemos utilizar una función (function). Las funciones operan con datos y variables en Python para realizar una acción deseada. Las funciones pueden tener tanto entradas como salidas, al igual que operadores matemáticos familiares como la suma, la resta, la multiplicación y la división (los cuales tienen cada uno dos entradas y una salida). Mientras que las funciones en Python pueden seguir teniendo un propósito matemático, como el cuadrado de un entero, Python permite un comportamiento más

abstracto de la función, como la impresión en la pantalla. En este caso, la función `print()` imprimirá su entrada a la pantalla.

Utilicemos las funciones matemáticas incorporadas de Python `sum()`, `min()`, y `max()` para calcular la estimación de éxito promedio de Jamaica, la estimación de éxito mínimo y la estimación de éxito máximo, respectivamente:

[24]:
```
nombre_de_pais = 'Jamaica'
lista_de_jamaica = estimados_de_exito[nombre_de_pais] # Lista de las estimaciones para␣
↪Jamaica
print(lista_de_jamaica)
```

```
[0.61, 0.65, 0.71]
```

[25]:
```
promedio_jamaica = sum(lista_de_jamaica) / len(lista_de_jamaica)
minimo_jamaica = min(lista_de_jamaica)
maximo_jamaica = max(lista_de_jamaica)
print("País:",nombre_de_pais,", Promedio:",promedio_jamaica)
print("País:",nombre_de_pais,", Mínimo:",minimo_jamaica)
print("País:",nombre_de_pais,", Máximo:",maximo_jamaica)
```

```
País: Jamaica , Promedio: 0.6566666666666666
País: Jamaica , Mínimo: 0.61
País: Jamaica , Máximo: 0.71
```

Fíjese arriba en cómo pasamos todo el objeto `lista_de_jamaica` como entrada a la función `min()`. La salida se asigna inmediatamente a una nueva variable que hemos llamado `minimo_jamaica`. También, note cómo podemos pasar más de un argumento a la función `print()`, separándolos con comas. Esto nos permite combinar datos dinámicos (variables) con cadenas de caracteres descriptivas y fijas para obtener salidas legibles. Encontraremos mejores formas de combinar datos variables en cadenas más adelante.

Como era de esperar, obtenemos el mismo resultado promedio de aproximadamente 0,657. Obsérvese que también podríamos haber redondeado los resultados a dos decimales utilizando el método round(). Esto puede mejorar la legibilidad.

[26]:
```
promedio_jamaica = round(sum(lista_de_jamaica) / len(lista_de_jamaica),2)
minimo_jamaica = round(min(lista_de_jamaica),2)
maximo_jamaica = round(max(lista_de_jamaica),2)
print("País:",nombre_de_pais,", Promedio:",promedio_jamaica)
print("País:",nombre_de_pais,", Mínimo:",minimo_jamaica)
print("País:",nombre_de_pais,", Máximo:",maximo_jamaica)
```

```
País: Jamaica , Promedio: 0.66
País: Jamaica , Mínimo: 0.61
País: Jamaica , Máximo: 0.71
```

Las funciones en Python son una herramienta muy poderosa para aumentar la productividad y realizar tareas más complejas.

Ejercicio 3 (5 min):

Escriba un *script (conjunto de líneas de código)* para calcular el promedio de éxito de cada país. Imprima (usando `print()`) la estimación del promedio de éxito de cada país en la pantalla. Las declaraciones de impresión deben dar salida a cada país en una nueva línea, por ejemplo:

```
País: Francia , Promedio: 0.655
País: Brasil , Promedio: 0.29
```

Respuesta.

Determinar sistemáticamente el promedio de éxito estimado para todos los países (15 min)

El objetivo final de este análisis es una recomendación sobre dónde deberían considerarse las oportunidades de expansión mundial. Para llegar a una conclusión, lo ideal sería tener la probabilidad media de éxito de cada país.

Para lograrlo, utilizaremos un elemento de flujo de control en Python - el bucle for. El bucle `for` permite ejecutar las mismas sentencias una y otra vez (es decir, iterando). Esto ahorra una cantidad significativa de tiempo en la codificación de tareas repetitivas y ayuda a la legibilidad del código. La estructura general de un bucle `for` es:

```
for variable_iteradora in alguna_secuencia:
    declaración(es)
```

El bucle `for` itera sobre `alguna_secuencia` y ejecuta `declaraciones` en cada iteración. Es decir, en cada iteración de la `variable_iteradora` se actualiza al siguiente valor en alguna secuencia. Como ejemplo concreto, considera el bucle:

```
for i in [1,2,3,4]:
    print(i*i)
```

Aquí, el bucle `for` se imprimirá en la pantalla cuatro veces; es decir, imprimirá "1" en la primera iteración del bucle, "4" en la segunda, "9" en la tercera y "16" en la cuarta. Por lo tanto, la sentencia del bucle `for` itera sobre todos los elementos de la lista `[1,2,3,4]`, y en cada iteración actualiza la variable iteradora `i` al siguiente valor de la lista `[1,2,3,4]`. En los bucles `for`, es un hábito extremadamente bueno elegir una variable iteradora que proporcione un contexto en lugar de una letra aleatoria. En este caso, usaremos ambas para acostumbrarle a ambas, ya que es muy probable que las vea a lo largo de su carrera en ciencias de datos, pero le animamos a no usar un nombre genérico como `i` siempre que sea posible para facilitar la comunicación.

Usemos un bucle `for` en los datos de nuestros países, obteniendo una lista de todas las llaves presentes en `estimados_de_exito`:

[28]:
```
# Obtener todas las llaves del diccionario de estimaciones de éxito
lista_nombres_paises = list(estimados_de_exito.keys())
print(lista_nombres_paises)
```

```
['Australia', 'France', 'Italy', 'Brazil', 'USA', 'England', 'Canada', 'Argentina',
'Greece', 'Morocco', 'Tunisia', 'Egypt', 'Jamaica', 'Switzerland', 'Germany']
```

Aquí haremos un bucle a través de todos los elementos de `lista_nombres_paises`, extraemos el valor correspondiente de `estimados_de_exito` (que será de tipo lista), y posteriormente tomamos el medio de la lista. La impresión detallada te guiará a través de la ejecución del bucle `for`:

[29]:
```
# Loop through all countries and calculate their mean success estimate
for i in lista_nombres_paises:
    print('--Comienza una iteración del bucle--')
    print('Elemento de la lista de nombres de países, marcador de posición i = ' + i)
```

```
    print('Acceder al valor de diccionario estimados_de_exito[i]: ',␣
↪estimados_de_exito[i])
    print('Promedio de la lista de estimados_de_exito[i]: ',␣
↪sum(estimados_de_exito[i]) / len(estimados_de_exito[i]))
    print('--Pasa a la siguiente iteración del bucle--')
    print()
```

```
--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Australia
Acceder al valor de diccionario estimados_de_exito[i]:  [0.6, 0.33, 0.11, 0.14]
Promedio de la lista de estimados_de_exito[i]:  0.29500000000000004
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = France
Acceder al valor de diccionario estimados_de_exito[i]:  [0.66, 0.78, 0.98, 0.2]
Promedio de la lista de estimados_de_exito[i]:  0.655
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Italy
Acceder al valor de diccionario estimados_de_exito[i]:  [0.6]
Promedio de la lista de estimados_de_exito[i]:  0.6
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Brazil
Acceder al valor de diccionario estimados_de_exito[i]:  [0.22, 0.22, 0.43]
Promedio de la lista de estimados_de_exito[i]:  0.29
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = USA
Acceder al valor de diccionario estimados_de_exito[i]:  [0.2, 0.5, 0.3]
Promedio de la lista de estimados_de_exito[i]:  0.3333333333333333
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = England
Acceder al valor de diccionario estimados_de_exito[i]:  [0.45]
Promedio de la lista de estimados_de_exito[i]:  0.45
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Canada
Acceder al valor de diccionario estimados_de_exito[i]:  [0.25, 0.3]
Promedio de la lista de estimados_de_exito[i]:  0.275
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Argentina
```

```
Acceder al valor de diccionario estimados_de_exito[i]:  [0.22]
Promedio de la lista de estimados_de_exito[i]:  0.22
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Greece
Acceder al valor de diccionario estimados_de_exito[i]:  [0.45, 0.66, 0.75, 0.99, 0.15,
0.66]
Promedio de la lista de estimados_de_exito[i]:  0.61
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Morocco
Acceder al valor de diccionario estimados_de_exito[i]:  [0.29]
Promedio de la lista de estimados_de_exito[i]:  0.29
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Tunisia
Acceder al valor de diccionario estimados_de_exito[i]:  [0.68, 0.56]
Promedio de la lista de estimados_de_exito[i]:  0.6200000000000001
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Egypt
Acceder al valor de diccionario estimados_de_exito[i]:  [0.99]
Promedio de la lista de estimados_de_exito[i]:  0.99
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Jamaica
Acceder al valor de diccionario estimados_de_exito[i]:  [0.61, 0.65, 0.71]
Promedio de la lista de estimados_de_exito[i]:  0.6566666666666666
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Switzerland
Acceder al valor de diccionario estimados_de_exito[i]:  [0.73, 0.86, 0.84, 0.51, 0.99]
Promedio de la lista de estimados_de_exito[i]:  0.7859999999999999
--Pasa a la siguiente iteración del bucle--

--Comienza una iteración del bucle--
Elemento de la lista de nombres de países, marcador de posición i = Germany
Acceder al valor de diccionario estimados_de_exito[i]:  [0.45, 0.49, 0.36]
Promedio de la lista de estimados_de_exito[i]:  0.4333333333333333
--Pasa a la siguiente iteración del bucle--
```

Veamos más de cerca el bucle `for` anterior. La `lista_nombres_paises` tiene 15 países sobre los cuales el bucle `for` se repite. El bucle `for` usa una variable de marcador de posición, denotada como `i` utilizada en este caso, para almacenar el elemento de `lista_nombres_paises` al que corresponde cada iteración del bucle. Es decir, para la primera iteración del bucle `for`, `i = 'Brasil'`. Para la segunda iteración, `i =`

`'Canada'`. Y así sucesivamente hasta que el bucle llega al elemento final de `lista_nombres_paises`, que luego completa y sale del proceso del bucle.

¿Por qué es útil este proceso de bucle? Bueno, ¡hemos realizado las mismas sentencias de cálculo 15 veces mientras sólo escribíamos el código una vez! Observe que para cada iteración, se accede al valor correspondiente de `success_estimate`, y se calcula la media de la lista devuelta. El proceso del bucle `for` también mejora la legibilidad del código.

Ejercicio 4 (4 min):

Escriba un bucle `for` para calcular el mínimo y el máximo de la lista de estimaciones de éxito de cada país, imprimiendo cada una consecutivamente como en el ejemplo del anterior bucle `for`.

Respuesta.

Ejercicio 5 (4 min):

Usando el bucle `for`, escriba el código para determinar el país con el mayor rango de estimaciones de éxito (es decir, la mayor diferencia entre la estimación más pequeña y la más grande para un país).

Respuesta.

Usando "list comprehensions" para determinar el número de estimaciones (12 min)

En el futuro, nos interesa saber el número de estimaciones de éxito disponibles para cada país. Python ofrece una manera concisa de lograr este objetivo mediante el uso de list comprehensions. Cada "list comprehension" también puede ser reescrita como un bucle `for`. La sintaxis de "list comprehension" es simplemente más concisa.

Las list comprehension permiten construir una lista de forma concisa. Demos un vistazo a cómo funciona esto:

[32]:
```
lista_nombres_llave = [i for i in estimados_de_exito] # Pasa por cada elemento i en␣
↪las estimaciones de éxito y lo coloca en la lista
lista_nombres_llave
```

[32]:
```
['Australia',
 'France',
 'Italy',
 'Brazil',
 'USA',
 'England',
 'Canada',
 'Argentina',
 'Greece',
 'Morocco',
 'Tunisia',
 'Egypt',
 'Jamaica',
 'Switzerland',
 'Germany']
```

Aquí vemos que hemos hecho un bucle sobre cada llave del diccionario `estimados_de_exito` (por lo tanto, cada país), y hemos extraído el nombre del país, todo en una línea de código. También podemos acceder a los valores de cada llave en `estimados_de_exito`:

[33]:
```
lista_valores = [estimados_de_exito[pais] for pais in estimados_de_exito] # hace un␣
↪bucle sobre cada elemento en "estimados_de_exito" y pone "estimados_de_exito[i]" en␣
↪la lista
lista_valores
```

[33]:
```
[[0.6, 0.33, 0.11, 0.14],
 [0.66, 0.78, 0.98, 0.2],
 [0.6],
 [0.22, 0.22, 0.43],
 [0.2, 0.5, 0.3],
 [0.45],
 [0.25, 0.3],
 [0.22],
 [0.45, 0.66, 0.75, 0.99, 0.15, 0.66],
 [0.29],
 [0.68, 0.56],
 [0.99],
 [0.61, 0.65, 0.71],
 [0.73, 0.86, 0.84, 0.51, 0.99],
 [0.45, 0.49, 0.36]]
```

En la "list comprehension" anterior, cada valor de la `variable_iteradora` es un nombre de país y el valor para cada país se obtiene cuando se llama `estimados_de_exito[pais]`. Vemos que la "list comprehension" es una forma efectiva y concisa de escribir un bucle `for` que crea una lista.

Podemos usar esto para determinar rápidamente cuántas estimaciones de éxito están disponibles para cada país:

[34]:
```
# Número de estimaciones disponibles para cada país
[[i, len(estimados_de_exito[i])] for i in estimados_de_exito]
```

[34]:
```
[['Australia', 4],
 ['France', 4],
 ['Italy', 1],
 ['Brazil', 3],
 ['USA', 3],
 ['England', 1],
 ['Canada', 2],
 ['Argentina', 1],
 ['Greece', 6],
 ['Morocco', 1],
 ['Tunisia', 2],
 ['Egypt', 1],
 ['Jamaica', 3],
 ['Switzerland', 5],
 ['Germany', 3]]
```

Ejercicio 6 (3 min):

Usando las "list comprehensions", escriba un script (conjunto de líneas de código) para crear una lista de listas llamada `lista_suma_cuadrados`, donde cada elemento de la lista es una lista de dos elementos [nombre del país, valor]. El elemento de valor de la lista debe ser la suma de los cuadrados de las estimaciones de éxito de ese país. Por ejemplo, el elemento de la `lista_suma_cuadrados` correspondiente a Jamaica debe ser [Jamaica, 1.2987], (ya que 1.2987 = 0,61^2 + 0,65^2 + 0,71^2).

Respuesta.

Ejercicio 7 (5 min):

Nos gustaría determinar la dispersión alrededor de la estimación media de éxito para cada país. Usando "list comprehensions", escriba un script (conjunto de líneas de código) que reste la estimación media de éxito para un país dado de cada estimación de éxito para ese país. Almacene los resultados en una lista llamada `lista_media_removida`. Redondee los valores a dos decimales. Su salida debe producir la siguiente lista de listas:

```
[[['Australia', [0.3, 0.03, -0.19, -0.16]],
 ['Francia', [0.01, 0.12, 0.32, -0.46]],
 ['Italia', [0.0]],
 ['Brasil', [-0.07, -0.07, 0.14]],
 ['USA', [-0.13, 0.17, -0.03]],
 ['Inglaterra', [0.0]],
 ['Canadá', [-0.03, 0.02]],
 ['Argentina', [0.0]],
 ['Grecia', [-0,16, 0,05, 0,14, 0,38, -0,46, 0,05]],
 ['Marruecos', [0.0]],
 ['Túnez', [0.06, -0.06]],
 ['Egipto', [0.0]],
 ['Jamaica', [-0.05, -0.01, 0.05]],
 ['Suiza', [-0.06, 0.07, 0.05, -0.28, 0.2]],
 ['Alemania', [0.02, 0.06, -0.07]]]
```

Respuesta.

A medida que la "list comprehension" se complica, se hace cada vez más importante utilizar convenciones de nombrado adecuadas para las variables. Las buenas convenciones de nombrado mejoran dramáticamente la legibilidad de su código.

Reflexionando sobre la estimación del éxito medio de cada país (5 min)

Basándonos en el análisis anterior, vemos que las estimaciones medias de éxito de los países varían ampliamente, desde la más baja, Canadá = 0,275, hasta la más alta, Egipto = 0,99. Sin embargo, observe que la media de Egipto se calcula a partir de 1 estimación de éxito. ¿Confiamos en una sola estimación como sustituto de la estimación media de éxito?

Dado que el proyecto de expansión mundial utilizará valiosos recursos de la empresa, decidimos que es mejor restringir nuestro análisis a los países que tienen dos o más estimaciones de éxito. Para llevar a cabo esta tarea, utilizaremos una estructura de control en Python conocida como condicionales if…elif…else. La estructura general es la siguiente:

```
if condicion_1:
    bloque_de_declaraciones_1
elif condicion_2:
    bloque_de_declaraciones_2
else:
    bloque_de_declaraciones_3
```

En este caso, `condicion_1` y `condicion_2` deben ser verdaderas (`True`) o falsas (`False`), y correspondientes al tipo (type) boolean de Python . El tipo booleano se asocia con variables que son verdaderas (`True`) o falsas (`False`).

Si `condicion_1` es verdadera (`True`), el `bloque_de_declaraciones_1` se ejecutará y las otras declaraciones de bloque no lo harán. Si `condicion_2` es Falsa (`False`) pero la `condicion_2` es Verdadera (`True`), entonces se ejecutará `bloque_de_declaraciones_2` y las otras no lo harán. Por último, si `condicion_1` y `condicion_2` son ambas falsas (`False`), entonces se ejecutará el `bloque_de_declaraciones_3`. Esta estructura condicional de la sentencia tipo `if` permite controlar el flujo de código Python.

Los bloques `elif` y `else` son opcionales. También puede tener un bloque `if` simple que solamente ejecuta algunas líneas de código, o las salta, dependiendo de la evaluación del condicional. Como ejemplo, veamos sólo dos países de nuestro conjunto de datos: Italia y Jamaica:

[37]:
```
italy = estimados_de_exito['Italy']
jamaica = estimados_de_exito['Jamaica']

print(len(italy) > 1)
print(len(jamaica) > 1)
```

```
False
True
```

Arriba, usamos el operador mayor que `>` para comparar dos números y producir un booleano (el tipo de datos de Python más simple - siempre verdadero o falso). En el primer caso, obtuvimos `False` porque la longitud de la lista `italy` es **no** mayor que 1. Dicho de otra manera, no es cierto que $1 > 1$, por lo que obtenemos `False`. Por el contrario, Jamaica tiene 3 ítems, por lo que la evaluación se simplifica a `3 > 1` que es Verdadera, por lo que en la segunda línea de arriba vemos la salida `True`.

Podemos combinar estas expresiones con declaraciones `if` para controlar el flujo de nuestro código. A continuación se muestra un ejemplo:

[38]:
```
if len(italy) > 1:
    print("Italy tiene más de un estimado")

if len(jamaica) > 1:
    print("Jamaica tiene más de un estimado")
```

```
Jamaica tiene más de un estimado
```

Arriba, podemos ver que se saltó la segunda línea del código. Nunca se ejecutó y no vemos ninguna salida para la declaración que está "contenida" allí.

Lo anterior muestra dos bloques de `if` separados. Note que hay una diferencia entre tener dos bloques de `if` separados y un bloque `if else`, como se muestra a continuación:

[39]:
```
# **Advertencia de lógica incorrecta**
if len(italy) > 0:
    print("Italy tiene más de 0 estimados")
```

```
elif len(jamaica) > 0:
    print("Jamaica tiene más de 0 estimados")
else:
    print("esto sólo se ejecutará si ninguna de las líneas anteriores lo hace")
```

```
Italy tiene más de 0 estimados
```

El ejemplo anterior muestra lo cuidadoso que hay que ser al codificar. Usamos `0` en lugar de `1` como la longitud mínima de la lista que nos interesa. Debido a que la primera declaración es cierta `True` (Italia *tiene* más de cero estimaciones), Python se salta el resto del código sin mirarlo. Aunque la siguiente condición para Jamaica es *también* verdadera `True`, el hecho de estar en un bloque `elif` nunca hará que se ejecute si alguna expresión anterior del mismo bloque es evaluada como verdadera `True`.

Verá un uso más avanzado de la lógica condicional usando `if` más adelante. Por ahora, combinemos las declaraciones `if` con los bucles `for` para filtrar todos los países que sólo tienen una estimación de éxito.

Seleccionando sólo los países con múltiples observaciones para el potencial de expansión global (15 min)

Utilizaremos la declaración del if anterior para eliminar a los países con menos de una estimación de éxito. Convenientemente, para ver el resultado, almacenaremos las estimaciones medias para cada país en un nuevo diccionario `media_pais`:

[40]:
```
# Obtiene una lista de todos los nombres de los países
lista_nombres_paises = list(estimados_de_exito.keys())

# Crea un diccionario vacío para guardar las estimaciones de las medias de los países
media_paises = {}

# Pasa por todos los países y calcula su promedio de éxito estimado
for i in lista_nombres_paises:
    lista_estimado_paises = estimados_de_exito[i] # lista de estimaciones para un país

    # si se estima más de un país, entonces registra la estimación media, si no, pasa␣
 ↪a la siguiente iteración del bucle
    if len(estimados_de_exito[i]) > 1:
        print("adicionando la media para ",i)
        valor_media_pais = sum(lista_estimado_paises) / len(lista_estimado_paises)
        media_paises[i] =  valor_media_pais # inserta el valor medio del país en el␣
 ↪diccionario usando el nombre del país como llave
    else:
        print("....saliendo ", i)
```

```
adicionando la media para  Australia
adicionando la media para  France
…saliendo  Italy
adicionando la media para  Brazil
adicionando la media para  USA
…saliendo  England
adicionando la media para  Canada
…saliendo  Argentina
adicionando la media para  Greece
…saliendo  Morocco
```

```
adicionando la media para  Tunisia
…saliendo  Egypt
adicionando la media para  Jamaica
adicionando la media para  Switzerland
adicionando la media para  Germany
```

Observe que se omiten los países que no tienen más de una estimación. No sólo se ignoran las líneas de impresión (`print`), sino que también se ignoran todas las líneas contenidas en el bloque `if`, así que las medias para los países con una sola estimación no se añaden a `media_paises`.

Ahora, demos formato a nuestros resultados, modificando la salida de cadenas de caracteres a la pantalla para usar 2 decimales cuando se imprima el tipo flotante. Esto se logra usando la funcionalidad `.format()` de los objetos tipo (type) cadena de caracteres (string). Los `{0:s}` y `{1:.2f}` en la cadena de caracteres (string) le indican al método `.format()` que debe formatear la primera variable de entrada que recibe como una cadena reemplazando el marcador de posición `{0:s}`; y formatear la segunda variable de entrada como un flotante de 2 decimales reemplazando el marcador de posición `{1:.2f}`.

Con este formato, la variable `llave_pais` se mostrará como un "string" en la posición de `{0:s}`, mientras que la variable `media_paises[llave_pais]` se mostrará como un flotante de 2 decimales la posición de `{1:.2f}`. Este enfoque avanzado de formato de "string" es útil para mejorar la claridad de los resultados:

[41]:
```
# Formato agradable del resultado para imprimirlo en la pantalla
for llave_pais in media_paises:
    print("País: {0:s}, Promedio de estimados de éxito: {1:.2f}".format(llave_pais,␣
 ↪media_paises[llave_pais]))
```

```
País: Australia, Promedio de estimados de éxito: 0.30
País: France, Promedio de estimados de éxito: 0.66
País: Brazil, Promedio de estimados de éxito: 0.29
País: USA, Promedio de estimados de éxito: 0.33
País: Canada, Promedio de estimados de éxito: 0.28
País: Greece, Promedio de estimados de éxito: 0.61
País: Tunisia, Promedio de estimados de éxito: 0.62
País: Jamaica, Promedio de estimados de éxito: 0.66
País: Switzerland, Promedio de estimados de éxito: 0.79
País: Germany, Promedio de estimados de éxito: 0.43
```

Observando las medias de los países resultantes, observamos que el país con la mayor estimación media de éxito es Suiza, con 0,79, mientras que la estimación media más baja de éxito es el Canadá, con 0,28.

Ejercicio 8 (6 min):

Después de revisar la política de la compañía sobre procedimientos estadísticos, notará que la compañía recomienda que todas las estimaciones (promedios, mínimos, máximos) deben tener por lo menos tres valores que contribuyan a la estadística de resumen. Escriba un bucle `for` y utilice la estructura de condicionales `if` para seleccionar e imprimir las estimaciones promedio de éxito para los países que satisfacen esta política. Si el país no satisface la política, imprima el nombre del país y `*No cumple con la política de la compañía*`. Cada país debe aparecer en una nueva línea.

Respuesta.

Ejercicio 9: (6 min)

Además de la política anterior, la empresa quiere señalar a los países en los que el rango de estimaciones es demasiado grande, ya que esto suele deberse a que las estimaciones se toman en momentos muy diferentes, o que contienen errores. Quieren prestar especial atención a los casos en que el rango (la diferencia entre las estimaciones más grandes y las más pequeñas) es mayor de 0,6. Calculen el rango para cada país e indiquen si debe ser marcado.

Respuesta.

Ejercicio 10 (5 min):

¿Cuál es otro enfoque para resolver el problema de la muestra única para algunos países? Piense en términos de los factores que impulsan la confianza en las decisiones comerciales basadas en datos.

a) Agrupar los países en regiones más grandes para asegurarse de que cada región tenga al menos una estimación

b) Retirar un país sólo si sus previsiones son muy grandes o muy pequeñas en comparación con otras previsiones

c) Utilizar una estadística resumida diferente para el análisis que no sea el valor medio

d) Volver a examinar por qué algunos países sólo tienen una estimación y ver si se pueden obtener más datos para esos países

Respuesta.

Uniendo todo (5 min)

Hemos utilizado bucles `for` y las estructuras `if` de control para calcular las estadísticas resumidas parciales de cada uno de los países. Pongámoslo todo junto para obtener una recomendación sobre qué país debemos elegir para ampliar los servicios de vuelos de lujo.

Ejercicio 11: (5 min)

Escriba el código para imprimir el nombre de cada país y el resumen de las estadísticas. Cada línea debe mostrar un país y las correspondientes estadísticas resumidas: Estimación mínima (float), estimación media (float), estimación máxima (float), número de estimaciones (int), cumple con la política de la empresa de al menos 3 estimaciones (bool).

Por ejemplo, la línea correspondiente a Francia aparecería como:

```
País: Francia , Min: 0,2 , Promedio: 0,655 , Máximo: 0,98 , NúmEst: 4 , Cumple con la política: Tr
```

Respuesta.

Ejercicio 12: (10 min)

Podemos ver que Suiza tiene el promedio más alto mirando todos los valores, pero demos un paso más y hagamos que Python nos lo calcule.

Vuelva a revisar los datos, pero haga un seguimiento del mejor país **que cumple con la política** y obtenga datos de salida sólo de este país.

Respuesta.

Ejercicio 13: (10 min)

Encontramos el mejor país en promedio, pero también estamos interesados en la elección más segura o más conservadora. Esto se define mirando sólo al **mínimo** (es decir, el más pesimista) estimado para cada país y tomando el más alto de aquellos (considerando sólo aquellos **que cumplen con la política**). Escriba el código para determinar y producir los datos sobre este país.

Respuesta.

Note que arriba usamos la palabra llave `and` de Python para combinar dos booleanos en uno. El resultado es verdadero (`True`) sólo si **ambos** componentes son verdaderos (`True`). Tiene el mismo efecto que la declaración anidada `if` del ejercicio 12. Puedes leer más sobre los operadores lógicos en el tutorial de W3Schools.

Antes de concluir, vamos a llevar todo nuestro trabajo a GitHub para mantenerlo seguro.

- Asegurese de que cualquier código y archivos del Notebook que tenga abierto se guarde.
- Abra su línea de comandos del sistema o aplicación terminal y navegue a su repositorio git (que también debería contener su trabajo)
- Ejecute `git status` para comprobar el estado de los archivos. Borre cualquier archivo que no quiera enviar a GitHub.
- Ejecute `git add *` para añadir todos los archivos
- Ejecute `Git commit" -m "Terminado el caso de la preparación de Python"`
- Ejecute `git push origin master`

En su terminal debería ver la salida confirmando que los archivos fueron sometidos, y si ingresa a github.com y navega a su repositorio, debería ver los archivos con los últimos cambios a través de la interfaz web.

Ahora que tenemos estadísticas resumidas para una variedad de estimaciones de países, construyamos nuestra recomendación de negocio.

Conclusiones (3 min)

Del análisis se desprende que Suiza es el país con mayores posibilidades de éxito para la expansión mundial, con una tasa de éxito estimada de 0,79. El resumen estadístico de Suiza se calculó con un número adecuado de estimaciones según la política de la empresa. Así pues, se recomienda a la dirección que explore las oportunidades en Suiza para los servicios de vuelos de lujo. También se recomienda que Jamaica sea una opción conservadora. Aunque en general no ocupó el primer lugar, ninguna de sus estimaciones tuvo una probabilidad considerablemente baja.

Además, otro país que debería ser objeto de una estrecha vigilancia en lo que respecta a los servicios de vuelos de lujo es Francia, que mantuvo una estimación de éxito media de 0,66. Si hay recursos adicionales en el futuro para una mayor expansión de los servicios de lujo, Francia puede ser una elección acertada, pero se debe investigar más para ver por qué una de sus estimaciones alcanzó un valor de solo 0,2.

Tenga en cuenta y lleve en mente (5 min)

En este caso, hemos aprendido los fundamentos de Python. A través de la identificación de oportunidades de expansión global para una compañía aérea hemos cubierto tipos de datos fundamentales, estructuras de control y un flujo de trabajo útil de Python para analizar un determinado conjunto de datos. También pudo

aprender sobre varias medidas estadísticas de resumen y que tanta confianza brindan al momento de sacar conclusiones a partir de ellas.

Para recapitular, estos son algunos de los componentes de Python que acabamos de cubrir. Le recomendamos que se familiarice con ellos antes de seguir adelante: - variables - tipos de datos - `print` (cómo dar formato dentro de la función `print`) - listas - diccionarios - deslizar(rebanar)/iterar - el bucle `for` - declaraciones condicionales `if...elif...else` - "list comprehension"

De aquí en adelante, puede utilizar estas herramientas de Python como base y marco para construir proyectos más complejos y resolver problemas críticos de negocio. Python sigue siendo una herramienta excepcional para realizar análisis basados en datos y proporcionar conocimientos empresariales clave.