SESION 1 FASE 3. PROCESAMIENTO DE DATOS CON PYTHON

Victor Miguel Terrón Macias

19/2/2021

FUNDAMENTOS DE PYTHON

OBJETIVO

- Conocer las dos principales estructuras de datos que son la base de la programación con Python.
- Comparar dos diferentes maneras de estructurar datos y saber las ventajas y desventajas de cada una.
- Reutilizar código y hacer nuestros programas más modulares.
- Entender mejor el funcionamiento detrás de las funciones built-in de Python.

INTRODUCCIÓN

En esta primera sesión aprenderemos a usar algunas herramientas sumamente importantes que estaremos utilizando a través de todo el módulo. Empezaremos platicando rápidamente sobre el Procesamiento de Datos y por qué es tan importante. Después, aprenderemos acerca de Jupyter Notebooks y por qué son útiles para los científicos de datos. Además, aprenderemos algunas cosas básicas del lenguaje de programación Python.

IMPORTANCIA DEL PROCESAMIENTO DE DATOS

Todo proyecto de Ciencia de Datos tiene una serie de pasos necesarios para ser realizado con éxito. Normalmente todo empieza con la identificación de un problema en el mundo real. Después de encontrar un problema que creemos que requiere una solución, el paso siguiente es entender el problema. Para entender el problema, necesitamos datos. Puede que ya exista una base de datos disponible que haya sido construida para recopilar instancias del proceso que estamos estudiando. Puede que esa base de datos aún no exista y primero necesitemos pensar sobre cómo recopilar esos datos. De cualquier manera, normalmente el proceso de recopilación y adquisición de los datos es un proceso algo confuso. La realidad no es ordenada, y nuestros datos van a ser un reflejo de esa realidad. Es muy probable que a la hora de obtener nuestra base de datos, nos demos cuenta de que está llena de errores: datos vacíos, formatos inadecuados, columnas innecesarias, tipos de datos incorrectos, la lista sigue y sigue y sigue. Si intentamos hacer análisis estadístico o visualizaciones de datos con una base de datos desordenada, nos toparemos con muchos problemas. Nuestros análisis serán imprecisos y lo más probable es que terminemos con conclusiones incorrectas. Aquí es donde entra el Procesamiento de Datos. El Procesamiento de Datos reúne toda una serie de herramientas que sirven para explorar, limpiar, transformar, ordenar y estructurar los datos de manera que puedan ser útiles para su posterior análisis y visualización. En este módulo aprenderemos cómo realizar estos procesos esenciales y nos prepararemos para poder realizar procesos aún más complejos en módulos posteriores.

SOFTWARE A UTILIZAR

Para realizar los ejercicios de este módulo vamos a usar Jupyter Notebooks, Google Drive y Google Colab. A continuación vamos a ver cómo realizar la preparación de estas herramientas para que puedas sacar el mejor provecho.

CREANDO UN ACCESO DIRECTO HACIA LOS DATASETS

Dado que este es un curso de Análisis de Datos, es obvio que vamos a estar usando muchos conjuntos de datos. Para poder acceder a ellos durante el módulo, lo primero que necesitas es una cuenta de Google Drive. Si aún no tienes una puedes ir a este link para crear una. Ya que hayas creado tu cuenta, puedes proseguir con los siguientes pasos. Los conjuntos de datos que vamos a usar se encuentran aquí. Accede al link para ir a la carpeta donde están guardados los datasets de la sesión. Verás algo como esto (puede ser un poco distinto):

En caso de que no hayas hecho login en tu cuenta de Drive, hazlo ahora usando la cuenta que creaste en un paso anterior. Es importante que accedas a la carpeta de Datasets y hagas login desde tu cuenta para poder realizar los pasos siguientes. Da click en la parte superior, donde está el nombre de la carpeta "Datasets". El siguiente menú se desplegará:

Ahora, lo único que vamos a hacer es agregar un Acceso Directo desde nuestro Drive a esta carpeta. Da click en "Add Shortcut To Drive" (o "Agrega Acceso a Directo a tu Drive") y verás esto:

Elige el lugar en tu Drive donde quieres crear el Acesso Directo y da click en "Add Shortcut" (o "Agregar Acceso Directo"):

Listo, ahora puedes acceder a esta carpeta "Datasets" desde tu Google Drive. Vamos a ver ahora cómo vamos a aprovechar esto desde Google Colab. Pero antes, hablaremos rápidamente de qué es un Jupyter Notebooks.

Jupyter Notebooks

Un Jupyter Notebook es lo que se llama un REPL (Read-Eval-Print Loop), que es un entorno de programación computacional interactivo. ¡Woah! Suena muy complicado. En realidad es bastante simple. Veamos cómo se ve uno:

Como puedes ver, un Jupyter Notebook (a partir de ahora voy a llamarles JN) es algo parecido a un editor de texto. Una diferencia importante es que los JN están divididos en celdas. Una celda es un contenedor que puede tener dentro texto o código (de Python, por ejemplo). En nuestra imagen, vemos que nuestro JN comienza con una celda de texto que contiene un título, texto simple e incluso una lista. Estas celdas de texto se llaman celdas tipo Markdown, porque Markdown es el lenguaje que se usa para darles estilo. Debajo de nuestras celdas de texto tenemos celdas de tipo código. En estas celdas escribimos código en un lenguaje de programación (en este caso Python), que podemos después "correr" para obtener un resultado. El resultado de una celda de código se "imprime" justo debajo de la celda (en el ejemplo puedes ver los resultados de la suma y la multiplicación "impresos" justo debajo de la celda que realizó las operaciones). ¿Ves? En realidad es bastante sencillo, ¿no es así? Nosotros vamos a correr Jupyter Notebooks en la nube desde una plataforma gratuita que ofrece Google llamada Google Colab. No tenemos que realizar ninguna instalación. Simplemente sigue los pasos siguientes y estarás listo.

Google Colab Leyendo los Datasets desde Google Colab Ve a este link para acceder a Google Colab. Primero vamos a aprender cómo acceder a la carpeta Datasets desde Google Colab. Da click en "File/New Notebook":

Acabas de crear un Jupyter Notebooks (no te preocupes, ya aprenderemos cómo usarlo). Vamos a aprender a conectar nuestro Notebook con Google Drive. Primero, da click en la carpetita que aparece en el menú de la izquierda:

Aparecerá algo como esto:

Da click en “Mount Drive” (o “Montar Drive”) y una celda como la siguiente aparecerá:

Sigue el “url” y haz login con tu cuenta de Google:

Una vez que hagas login, verás una pantalla como ésta:

Copia el código y luego regresa a tu Jupyter Notebook. Pega el código en donde te piden que lo hagas y pulsa “Enter”. Listo:

Ahora, en el menú de la izquierda pica “Refresh” o “Refrescar” y verás tu Drive montado:

Si vas a la ruta donde creaste tu Acceso Directo, ¡puedes ver que todos los datasets están disponibles desde ahí!

Esto todavía no te va a hacer mucho sentido, pero si yo quisiera leer un dataset desde mi programa, sólo tendría que hacer algo como esto:

Ya aprendemos a leer archivos más adelante, pero lo importante es que sepas que ya tienes acceso desde Google Colab a todos los conjuntos de datos del módulo. Cada vez que abras un nuevo Jupyter Notebooks, tendrás que volver a realizar el montado de tu Google Drive.

ABRIENDO LOS RETOS DEÑ MÓDULO DESDE GOOGLE COLAB

Vamos a ver ahora cómo vamos a abrir los Retos del módulo usando Google Colab. Es muy sencillo. Primero que nada, necesitas el link del repositorio donde están guardados todos los Retos. El link es el siguiente:

LIGA

Ahora, en Google Colab, da click en “File/Open Notebook”:

Verás algo como esto:

Da click en la pestaña de Github, pega el link del repositorio y da “Enter”. Verás algo así:

Ahora puedes leer los archivos del repositorio desde Google Colab. ¡Genial! Para todos los ejercicios que hagas, asegúrate de elegir la branch “student”, para que tengas acceso a los archivos correctos:

Ahora, simplemente tienes que elegir el archivo que quieras leer. Google Colab sólo puede leer archivos de tipo .ipynb. Por ejemplo, si quiero acceder al primer Reto de la Sesión 1, tendría que abrir este archivo:

Si doy click, el Reto se abre y estoy listo:

Ahora que ya sabes cómo acceder a los conjuntos de datos y a los Retos, vamos a aprender a usar Jupyter Notebooks. IMPORTANTE: Los ejemplos de este módulo no fueron realizados en Google Colab. Se usaron Jupyter Notebooks corriendo localmente. Es por eso que las imágenes pueden verse algo distintas de lo que verás en Google Colab. El funcionamiento es idéntico, así que no te preocupes por eso. Usando **Jupyter Notebooks** Ya hablamos acerca de la estructura de un JN, vamos a poner todo esto en práctica. Esto en una celda:

Si presionamos return en nuestro teclado, entraremos al modo edición, donde podemos realizar cambios a esa celda. Observa que el borde de la celda cambia a color verde:

Para salir del modo edición, presionamos la tecla esc. Ahora estamos en lo que se llama modo comandos, y de indica con un borde azul alrededor de la celda:

Actualmente, nuestra celda es una celda de código, si queremos cambiarla a una celda markdown (de texto) presionamos las teclas cmd + b + m (para Mac, ahorita vemos para Windows) mientras estamos en el modo comando:

Todos los shortcuts en Google Colab empiezan con las siguiente combinación de teclas:

- Para MAC CMD+B, Para WINDOWS Ctrl+B

Escribes esas dos teclas y luego el shortcut correspondiente. Ahora podemos escribir algo en nuestra celda. Las celdas markdown reconocen un lenguaje especial llamado Markdown que nos permite agregar estilos a nuestro texto usando algunos signos muy simples. (si quieres aprender más sobre lenguaje Markdown, puedes revisar este link: Markdown Cheat Sheet) Presiona return para entrar en modo edición y teclea lo siguiente:

Los signos de numeral (#) sirven para escribir títulos en lenguaje Markdown. Ahora, tenemos que "correr" la celda para que el lenguaje muestre los estilos que hemos definido. Para correr celdas en JN se usa el comando shift + return. Mira lo que pasa después de correr la celda:

Ahora vamos a utilizar una celda de código. Para crear una nueva celda, entra en modo comandoy presiona cmd + b + b para crear una celda justo debajo de la que está seleccionada. Las celdas nuevas se crean siempre siendo celdas de código, pero en el caso de que quieras convertir una celda markdown a celda de código puedes hacerlo entrando en modo comando y presionando la tecla cmd + b + y. Ahora, habiendo seleccionado una celda de código, entra en modo edición y escribe lo siguiente:

No importa que no entiendas el código aún. Lo importante es saber que para correr esta celda, también tenemos que presionar shift + return. Al correr la celda, podemos ver el resultado de nuestras operaciones debajo de la celda que acabamos de correr:

¿Ves el número que sale a la izquierda de la celda (In [1]:)? Eso es un contador que nos va diciendo cuántas celdas hemos corrido en nuestra sesión actual. Cada vez que corres una celda de código, el contador de esa celda tomará el valor del último contador + 1:

Si queremos volver a iniciar nuestro JN desde cero, podemos ir al menú y elegir la opción Kernel/Restart & Clear Output:

Ésta ha sido una rápida introducción a Jupyter Notebooks. Por supuesto queda mucho por aprender. A través del módulo irás entendiendo cada vez mejor cómo es que funciona este REPL tan útil. Mientras tanto, estamos listos para empezar a aprender el lenguaje de programación Python. ¡Sigamos adelante!

Introducción a Python

En este prework vamos a hablar rápidamente acerca de algunos de los conocimientos fundamentales que necesitamos para programar en Python. Durante la clase presencial se realizarán numerosos ejercicios prácticos que te ayudarán a dominar estos temas. Por lo pronto, recomiendo ampliamente que vayas reescribiendo todo el código que veas aquí en tu propio JN. NUNCA hagas copy-paste. La única forma de aprender de verdad es escribir línea por línea, letra por letra, con tus propias manos.

Variables en Python

Los lenguajes de programación suelen incluir las famosas variables. Podemos pensar en las variables como "contenedores" de información a largo plazo. Son lugares en tu código donde puedes guardar cosas que quieres volver a utilizar después. Para poder crear una variable lo primero que tienes que hacer es elegir un nombre para la variable:

Después usas un operador de asignación que es un signo de "igual":

Y acto seguido, escribes el valor que quieres "asignar" a tu variable:

Entonces, el operador de asignación lo que hace es indicarle a Python que "el valor que está a la derecha del operador va a ser asignado a la variable que está a la izquierda del operador". Si corro mi celda, la variable ha sido asignada, y ahora ese número 10está "guardado" en mi variable variable_1. ¿Cómo accedo a ella? Basta con escribir el nombre de la variable y JN me muestra el valor como un output de mi celda:

OPERACIONES MATEMÁTICAS

Los operadores matemáticos en Python son exactamente los mismos que hemos aprendido en la escuela: +, -, *, /. Podemos usarlos para realizar operaciones matemáticas con nuestras variables (ya que nuestras variables tienen números asignados):

JN sólo me muestra el output de la última línea de código que hay en mi celda, así que si quiero ver el resultado de varias operaciones en la misma celda, tengo que usar un "comando" llamado print. print imprime lo que sea que reciba, para que podamos verlo con nuestros propios ojos:

Para usar el comando print escribimos la palabra "print", después abrimos paréntesis () y escribimos dentro del paréntesis lo que queremos imprimir. Estos "comandos" en Python se llaman funciones y es así como les llamaremos a partir de ahora. Veremos más a detalle lo que son las funciones en una sesión posterior.

TIPOS DE DATOS

En Python y en cualquier otro lenguaje de programación podemos representar diferentes tipos de datos. Hasta ahora hemos trabajado solamente con números. Pero de hecho Python no les llama "números". Python tiene dos tipos de datos para representar distintos tipos de números:

- *int* para números enteros
- *float* para números decimales
- *string* secuencia de caracteres para representar un texto
- *bool* VERDADERO O FALSO

Podemos usar la función type para "preguntarle" a Python qué tipo de dato tenemos en nuestras variables:

Para escribir texto en Python, tenemos que delimitar una secuencia de carácteres con el signo de comillas ("). Podemos imprimir strings para ir anotando los resultados de nuestro código:

También podemos imprimir variables dentro de strings usando lo que se llama interpolación de strings. Eso funciona de la siguiente manera: Empezamos una string con la letra f antes de las comillas, luego abrimos las comillas, después escribimos texto y cuando queremos interpolar una variable, usamos llaves ({}) y escribimos el nombre de la variable dentro de las llaves; finalmente cerramos las comillas:

¿Por qué queremos hacer comparaciones? Pues para tomar decisiones. ¿Y cómo tomamos decisiones en código? Utilizando las llamadas estructuras de control de flujo:

ESTRUCTURAS DE CONTROL DE FLUJO

Las estructuras de control de flujo nos permiten usar comparaciones para tomar decisiones acerca de qué pasos tomar en el futuro. La primera estructura de control es lo que llamamos if:

Le pido a Python que haga la comparación $4 < 7$, y si esta comparación resulta True entonces corro el código que tengo debajo. Presta atención al código de la segunda línea. Está indentado, ¿lo ves? Esto es muy importante. Para Python, todo el código que está indentado pertenece al bloque de la sentencia if y se correra si la comparación resulta ser True.

Podemos también agregar una condición default para decirle a Python: Si la comparación no se cumple, entonces corre este otro código. Esto lo hacemos usando el operador else:

También podemos encadenar varias comparaciones, para cuando queramos tener múltiples condiciones que resulten en diferentes acciones. Esto se hace usando el operador elif:

WORK SESION 1. FUNDAMENTOS DE PYTHON

OBJETIVOS

- Asiganr variables, operadores matemáticos y tipos de datos.
- Realizar interpolación de *strings*
- Realizar comapraciones y estructuras de control de flujo

CONTENIDO

Revisión de software Asegurarse de que todos los alumnos hayan realizado con éxito la conexión entre Google Drive, Github y Google Colab. Es necesario saber leer archivos .ipynb en Colab desde el repositorio del módulo. También es necesario haber creado el Acceso Directo a los Datasets desde el Drive del alumno para poder acceder a los conjuntos de datos desde Colab.

RETO 1. ASIGANCIÓN DE VARIABLES

```
# Asigna el número 12345 a una variable llamada variable_lococho
variable_lococho=12345
# Tu código va aquí...

# Asigna el número 14.567 a una variable llamada var_decimal
var_decimal=14.567
# ...

# Asigna cualquier número a una varible que tenga 5 palabras
# en su nombre (recuerda las convenciones)
vari_terron=3042310
# ....

# Asigna cualquier número a una variable que incluya palabras
# y números en su nombre
var_terron1=3042310
# ...

# Asigna variables a discreción
discrecion=var_terron1
# ...
```

Hay reglas para los nombres de las variables:

- No usar símbolos especiales como, !,@,#,%,etc.
- El primer caracter no puede ser un número
- Las constantes son colocadas dentro de modulos Python

RETO 2. OPERACIONES MATEMÁTICAS

Encuentra los valores:

```
var_1=4
var_2=10
var_3=6
var_4=8
var_5=2
#Se plantea a manera de ecuaciones, en un principio no se conoce el valor
#de las var_X
var_1+var_2
```

```
## 14
```

```
var_3-var_2
```

```
## -4
```

```
var_3*var_1
```

```
## 24
```

```
var_5+var_3
```

```
## 8
```

```
var_4/var_5
```

```
## 4.0
```

RETO 3. INTERPOLACIÓN DE STRINGS

OBJETIVOS

- Practicar interpolación de strings

DESARROLLO

A continuación tienes algunas variables asignadas:

```
nombre_1 = "Jenny"
nombre_2 = "Marco P."

edad_1 = 110
edad_2 = 42

estrella_1 = "Woopsie Doopsies"
estrella_2 = "Omega-3"
estrella_3 = "Justin Bieber"
estrella_4 = "La Twinkle"
estrella_5 = "Rosaberta"

total_de_nombres_de_estrellas_mencionados = 5
numero_de_estrellas_deseadas = 325
```

Usando **TODAS** las variables has que el texto consiga lo siguiente:

"Marco P. y Jenny son grandes amigues. Les mejores. Van y vienen juntes desde hace décadas. Sí, décadas. Marco P. tiene 110 años y Jenny 42. Si sumas sus edades, obtienes 152, que curiosamente es el número de estrellas que Marco P. y Jenny han nombrado juntes. Sus estrellas son su mayor fascinación. Las compraron en Best Buy en una ganga de objetos celestes. Están 'Woopsie Doopsies', 'Omega-3', 'Justin Bieber', 'La Twinkle', 'Rosaberta' y otras 147 estrellas más. Marco P. y Jenny tienen planeado sobrevivir hasta el siglo XXII. ¡Cuántas estrellas más habrán de comprar y nombrar juntes! Su meta es 325. Sólo faltan 173. ¡Vamos Marco P. y Jenny! Marco P. y Jenny son les mejores amigues. No hay duda alguna."

```
print(f"{nombre_1} y {nombre_2} son grandes amigues. Las mejores.")
```

```
## Jenny y Marco P. son grandes amigues. Las mejores.
```

```
print(f"Van y vienen juntos desde hace décadas. Sí, décadas. {nombre_1} tiene")
```

```
## Van y vienen juntos desde hace décadas. Sí, décadas. Jenny tiene
```

```
print(f"{edad_1} años y {nombre_2} {edad_2}. Si sumas sus edades, obtienes {edad_1 + edad_2},")
```

```
## 110 años y Marco P. 42. Si sumas sus edades, obtienes 152,
```

```
print(f"que curiosamente es el número de estrellas que {nombre_1} y {nombre_2} han nombrado juntos.")
```

```
## que curiosamente es el número de estrellas que Jenny y Marco P. han nombrado juntos.
```

```
print(f"Sus estrellas son su mayor fascinación. Las compraron en Best Buy en una ganga de objetos celes
```

```
## Sus estrellas son su mayor fascinación. Las compraron en Best Buy en una ganga de objetos celestes.
```

```
print(f" Están {estrella_1}, {estrella_2}, {estrella_3}, {estrella_4}, {estrella_5} y otras")
```

```
##  Están Woopsie Doopsies, Omega-3, Justin Bieber, La Twinkle, Rosaberta y otras
```

```
print(f"{edad_1 + edad_2 - total_de_nombres_de_estrellas_mencionados} estrellas más. {nombre_1} y {nomb
```

```
## 147 estrellas más. Jenny y Marco P.
```

```
print(f" tienen planeado sobrevivir hasta el siglo XXII. ¡Cuántas estrellas más habrán de comprar y nom
```

```
##  tienen planeado sobrevivir hasta el siglo XXII. ¡Cuántas estrellas más habrán de comprar y nombrar
```

```
print(f"Su meta es {numero_de_estrellas_deseadas}. Sólo faltan {numero_de_estrellas_deseadas - edad_1 +
```

```
## Su meta es 325. Sólo faltan 257
```

```
print(f". ¡Vamos {nombre_1} y {nombre_2}! {nombre_1} y {nombre_2} son les mejores amigues. No hay duda a
```

```
## . ¡Vamos Jenny y Marco P.! Jenny y Marco P. son les mejores amigues. No hay duda alguna.
```

RETO 4. OPERADORES DE COMPARACIÓN

OBJETIVOS

- Practicar la comparación de números, strings y booleanos

DESARROLLO

A continuación tienes algunas variables que han sido asignadas:

```
var_1 = 3
var_2 = 5
var_3 = 9
var_4 = 1
var_5 = 9
var_6 = -6
var_7 = 5
var_8 = 4
```

A continuación tienes algunas comparaciones que se encuentran incompletas, junto con el resultado que se espera de dicha comparación. Agrega los operadores de comparación adecuados, en medio de las variables, para que al correr las celdas se obtengan los Resultados esperados. Para este reto, los operadores == y != no están permitidos.

```
var_1 >var_3  # Resultado esperado: False
```

```
## False
```

```
var_1 >var_4 # Resultado esperado: True
```

```
## True
```

```
var_2 >=var_7  # Resultado esperado: True
```

```
## True
```

```
var_8 >var_6  # Resultado esperado: True
```

```
## True
```

```
var_3 >var_5  # Resultado esperado: False
```

```
## False
```

```
var_1 <var_3  # Resultado esperado: True
```

```
## True
```

```
var_5 <var_4  # Resultado esperado: False
```

```
## False
```

```
var_3 <=var_5  # Resultado esperado: True
```

```
## True
```

RETO 5. ESTRUCTURAS DE CONTROL DE FLUJO

OBJETIVOS

- Entender el funcionamiento de las estructuras de control de flujo

DESARROLLO

COMPARACIONES NUMÉRICAS

Vamos a imaginar que estamos analizando un conjunto de datos que contiene las ventas anuales en pesos de unos vendedores que trabajan en nuestra empresa. Obtenemos el total de ventas anuales en pesos y lo tenemos asignado a la variable ventas_anuales_totales. Escribe una estructura de control que nos diga de cuánto va a ser el bono de nuestro empleado.

Si las ventas son mayores a $1,000,000 el bono será de 20% del sueldo; si son entre $700,000 y $1,000,000 el bono será de 15% del sueldo; entre $400,000 y $700,000 el bono será de 10% del sueldo; entre $100,00 y 400,000 el bono será de 5% del sueldo; y debajo de $100,000 el bono será del 1% del sueldo.

Escribe tu estructura de control y, por ahora (más adelante tendremos más herramientas para hacer esto más interesante) imprime una string que diga de cuánto va a ser el bono.

Reto opcional: Agrega una variable sueldo que sea el sueldo de tu empleado y agrégale el porcentaje de bono a esa variable dependiendo de la condición que se cumpla. Después hasta el final imprime una string interpolada que diga algo como "Felicidades, tu sueldo es x; tu bono es de x; y el total de sueldo con el bono incluido es de x". ¡Para que esto funcione también debes de agregar una variable donde guardes la cantidad del bono!

```
ventas_anuales_totales =1000001 # Este valor lo puedes cambiar para obtener diferentes resultados

# OPCIONAL
sueldo =200 #escribe aquí el suelo de tu empleado
if ventas_anuales_totales>1000000:
  bono=sueldo*0.2
  print(bono)
elif ventas_anuales_totales>=700000 and ventas_anuales_totales<=1000000:
  bono=sueldo*0.15
  print(bono)
elif ventas_anuales_totales>=400000 and ventas_anuales_totales<700000:
  bono=sueldo*0.1
```

```
  print(bono)
elif ventas_anuales_totales>=100000 and ventas_anuales_totales<400000:
  bono=sueldo*0.05
  print(bono)
else:
  bono=salario*0.01
  print(bono)
```

```
40.0
```

```
print(f"Felicidades tu salario es de {sueldo + bono}.")
```

```
Felicidades tu salario es de 240.0.
```