Práctica 0 EDA II

Actividad 1. Si No se ha utilizado el Jupyter-notebook leer lo siguiente e ir realizándolo, en caso de haberlo utilizado pasar a la actividad 2

El entorno de programación Python es un lenguaje de programación de alto nivel, es interpretado, con características de orientación a objetos, con tipos dinámicos, e imperativo pero con algunas características de programación funcional. Más allá de los ejercicios de clase, podemos aprender más sobre el lenguaje, sus bibliotecas y sus diferentes versiones en el sitio Web oficial del lenguaje: www.python.org.

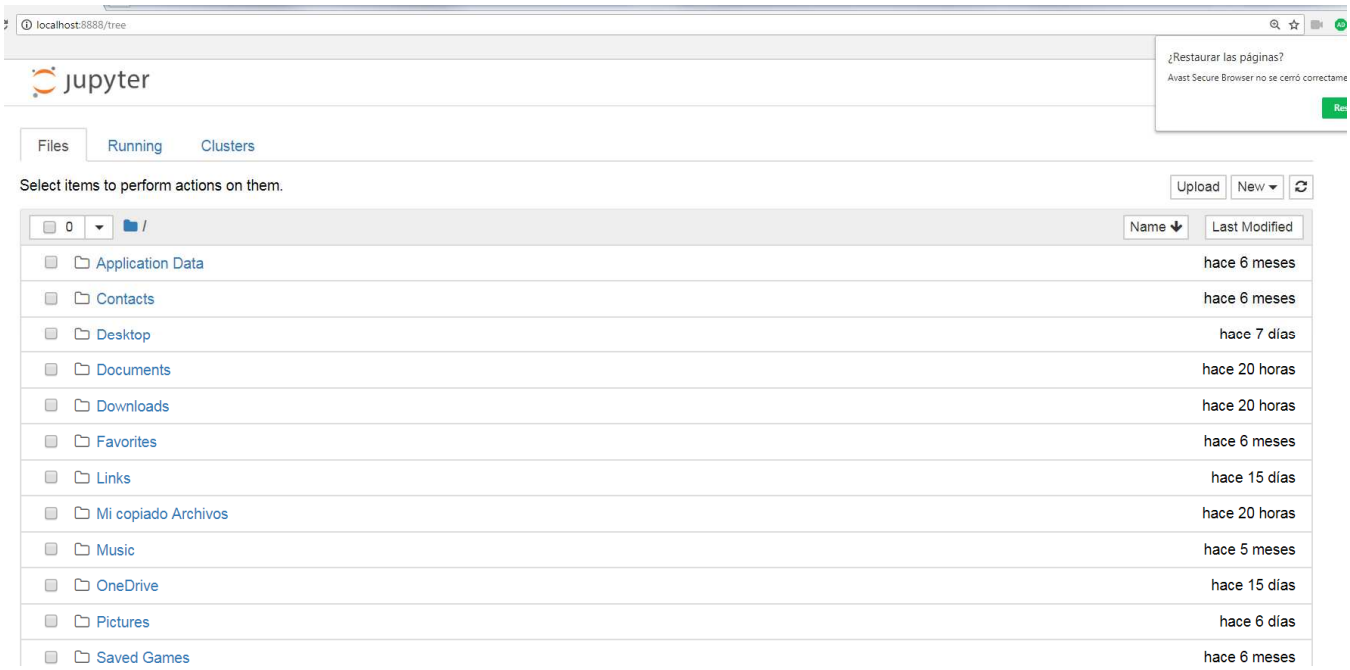
1. Para programar en Python necesitamos un editor de textos y el intérprete del lenguaje. En el laboratorio contamos con la herramienta Jupyter Notebook de Anaconda que nos proporciona un ambiente amigable.

   Parara utilizarlo en Windows, buscamos en la lista de programas Jupyter Notebook del botón Inicio/Windows. En el laboratorio es a veces es necesario abrir una ventana de línea de comandos, situarnos en una ruta donde tengamos permisos y teclear jupyter-notebook
   Si trabajamos en Linux, abrimos una ventana del terminal y ejecutamos:
   $ jupyter-notebook

   En ambos casos, se inicia un servidor local que nos permite utilizar la aplicación a través del navegador web. Si no se abre automáticamente la aplicación ingresamos a la siguiente dirección:

   http://localhost:8888 .

   En la pantalla principal vemos los archivos y carpetas de nuestra computadora.



Navegamos a una carpeta donde queramos guardar nuestros ejercicios y creamos un nuevo Notebook (New → Notebook → Python [Root])

Un Notebook es una secuencia de celdas, cada celda puede ser de distintos tipos, para el laboratorio nos interesan los tipos Markdown y Code. Las celdas Markdown nos sirven para escribir texto con formato (negritas, cursivas, títulos, etc.). El sistema de marcado es similar a HTML, y es el mismo que se utiliza en GitHub. El marcado es sencillo, podemos aprender sus comandos en

https://help.github.com/articles/basic-writing-and-formatting-syntax/

```
# Práctica 1
# Nombre del alumno

Ejemplo de celda _Markdown_.

- Podemos usar listas y
- enumeraciones

También podemos dar formato a nuestro texto:
1. Negritas: **Texto en negritas**, __Negritas usando 2 underscores__
2. Itálicas: *Texto en itálicas*, _También con 1 underscore_
3. Usamos una tilde inversa para texto que indique código dentro de una oración: `print("Hola Mundo")`
4. Usamos tres tildes inversas para bloques de código:
```
a = 1
b = 2
print(a + b)
```

Podemos estudiar la documentación del lenguaje de marcado en la [documentación de GitHub]
(https://help.github.com/articles/basic-writing-and-formatting-syntax/).
```

Las celdas de código nos permiten capturar y ejecutar código directamente en el documento:

```
In [2]: a = "Hola Mundo"
        x = 4
        y = 5
        print(a, "x * y: ", x*y)

        Hola Mundo x * y:  20
```

Para ejecutar código se puede utilizar la barra de herramientas que aparece en la parte superior y el botón  o la tecla shift + enter.

Actividad 2 Recordar algunas partes del Lenguaje

Los identificadores de variables en Python comienzan con una letra, underscore y están seguidas de cualquier letra, número y undersocres.

```
a = “variable tipo string”
valorInicial = 1  # variable tipo entero y un comentario
```

El uso de underscores al inicio puede tener un significado especial para el lenguaje. Se recomienda evitarlos.

2. Los tipos numéricos son como en otros lenguajes: enteros y punto flotante. En Python se incluye un tipo para números complejos:

```
a = 1 # entero
b = 3.4 # punto flotante
 c = 2+ 3j # número complejo
print(a, b, c)
```

3. Las constantes de cadenas pueden escribirse entre “ ” y ‘ ’. Podemos escribir cadenas que abarquen más de un reglón entre “”” y “””

```
s1 = "Esto es una cadena con retorno de carro₩n"
s2 = 'También podemos usar las "comillas sencillas"₩n'

s3 = """ Esta cadena
 abarca varios
 renglones … """
 print(s1, s2, s3)
```

4. Las listas en Python se escriben entre corchetes con sus elementos separados por comas.

```
a = [1, 3, 25]
```

Probar los siguientes ejemplos sobre las listas

```
a = [1, 3, 25]
b = [1] * 10
c=list()
d=[]
print(a)
print(b)
print(c)
print(d)
```

5. Podemos usar la función range() para crear listas:

```
a = list(range(4))
 b = list(range(5,10))
c = list(range(0, 100, 10))
 d = list(range(10, 0, -1))
print(a, b, c, d)
```

En todos los casos recuerda que el extremo derecho no es inclusivo (`range(5,10)` genera una secuencia de 5 a 9, y `range(10, 0, -1)` genera una secuencia de 10 a 1).

6. Usamos también los corchetes para indexar elementos y porciones de una lista:

```
 a = list(range(1,100))
print(a[0], a[1], a[10], a[25], a[50])
print(a[:10])
print(a[90:])
 print(a[25:50])
```

Observa que cuando usamos : para extraer rangos, el extremo derecho tampoco es inclusivo.

7. Podemos tomar decisiones con las sentencias if:

```
a = 5
 if a < 10:
    print("menor a 10")
elif a > 10:
      print("mayor a 10")
else:
       print("igual a 10")
```

Para definir los bloques de código que se ejecutan en cada sección del if, es necesario indentar el código. Cuando utilices un editor de textos, asegúrate que no utilice el carácter tabulador.

8. Python cuenta con ciclos for y while:

```
lista = [1,2,3]
contador = 0
for i in lista:
    print(i)
for i in range(0,11):
    print(i)

for i in range(10, -1, -1):
    print(i)

while contador <= 10:
    print(contador) contador += 1
```

9. Podemos definir funciones con la palabra clave def:

```
def cuadrado(n):
    return n*n

def factorial(n):
    if n <= 1:
        return n
    else:
        return n * factorial(n-1)

print(cuadrado(5))
print(factorial(5))
```

10. Si para leer una cadena, un entero y un flotante de la entrada estándar se puede realizar de la siguiente manera

In [ ]:
```
print ("Dame una cedena")
cadena = input ()
print ("La cadena es,",cadena)

entero = int(input("Introduce un entero: "))
print (entero)

flotante = float(input("Introduce un flotante : "))
print(flotante)
```