Guía para práctica de programación en Python

O. David Hincapie

Programación estructurada y métodos numéricos
Departamento de Ingeniería Mecánica
Universidad de Antioquia

1 Manejo de datos en Python

El manejo de datos es una habilidad que cualquier programador debe tener a la hora de realizar proyectos que involucren la programación. Durante la planeación y desarrollo de una aplicación, el programador define variables a través de estructuras de datos y los enlaza mediante sentencias y comandos.

A través de estas estructuras se crean objetos llamados datos que contienen información necesaria durante la ejecución de una aplicación. Estos objetos se almacenan en la memoria RAM ocupando un espacio determinado. El espacio que se necesita para guardar un objeto en memoria RAM depende del tipo de dato. Adicionalmente, la cantidad de espacio que se necesite en memoria RAM depende de la cantidad de variables que se creen. En algunos casos, se pueden llegar a tener códigos que permitan leer archivos grandes (del orden de GB, PB y TB) para realizar procesos más complejos en el mundo de la computación en cuestión de segundos.

Un dato puede tener tres posibles estados en un algoritmo:

1. **Creación:** Los objetos en cualquier lenguaje de programación se crean como "nombre_variable = valor". Es importante usar buenas prácticas de programación cuando se definen variables. Cuando se desee crear una o más variables, lo recomendable es se creen utilizando mismo idioma, esto incluye que no comience con números, que no tengan signos de puntuación ni cualquier otro signo y que la variable se separe por un identificador. Algunos ejemplos son:
   - number_of_iterations = 4
   - message_for_students = "Hay que ir a los talleres y a las asesorías"
   - finish_execution = False.
   - FinishExecution = False
2. **Manipulación:** Las variables se crean para interactuar con comandos y otras variables. Se puede definir la variable x = 5 y con esta variable hacer operaciones matemáticas, e incluso representaciones gráficas. La utilidad de la variable es definida por el programador, así como la interacción que tendrá esta variable con un código.
3. **Eliminación:** Para eliminar una variable, se usa el comando "del()". Dentro de los paréntesis, se indican aquellas variables que se deseen eliminar. Cuando se tengan variables que no se usen mucho en un algoritmo o cuando una variable ha cumplido su finalidad, se puede optar por borrar la variable para liberar espacio en memoria RAM.

En el siguiente código, se importa el modulo "os" para crear dos variables. Este módulo permite gestionar información del sistema operativo. En el comando usado en la linea 3 (getcwd()), se obtiene la carpeta donde esté contenido el archivo de python que contiene este código. El comando usado en la linea 4 (listdir()) retorna una lista de cadenas de caracteres con los nombres de los archivos que están en la carpeta por defecto.

Ejemplo de creación, manipulación y eliminación de variables

```
import os

path = os.getcwd()
objets_of_folder = os.listdir()
string_to_print = "La carpeta \t {} \t {} posee mas
    de 10 objetos"
if len(objets_of_folder)>0:
  print(string_to_print.format(path,'si'))
else:
  print(string_to_print.format(path,'no'))

del(path,objets_of_folder)
```

Se define un string que se imprime posteriormente según la cantidad de datos que contenga la carpeta. En este ejemplo, objects_of_folder permite decidir qué hará posteriormente el algoritmo, y string_to_print es modificada para luego ser mostrada en consola dependiendo del cambio que se hizo. Finalmente, se borran todas las variables creadas.

1.1 Tipos de datos

Los datos en la programación son una representación simbólica y digital de un atributo, estado o una variable cuantitativa y cualitativa. Existen áreas del conocimiento enfocadas a realizar estudios específicos en los datos. Algunos de estos involucran su almacenamiento, procesamiento, adquisición, minería, visualización, entre otros. Los principales tipos de datos son:

- **Simples:**
  - Entero (int)
  - Real (float)
  - Complejo (complex)
  - Carácter (str)
  - Lógico (bool)
- **Estructurados:**
  - Cadenas de caracteres (str)
  - Arreglos (array: numpy.ndarray)
  - Registros (tuple, list, dict, DataFrame)

Es importante que las sentencias que se definan e involucren tratamiento de datos tengan evalúen datos de la misma naturaleza. Existe una excepción involucrada a aquellos objetos relacionados con numpy que permiten operar arrays con datos escalares. Por ejemplo, el siguiente script define dos variables simbólicas a través del módulo scipy. Posteriormente, se usan estas variables para realizar integrales y solucionar la ecuación diferencial lineal de primer orden

$$y' + p(x)y = q(x)$$

Cuya solución se determina matemáticamente como

$$y(x) = \exp^{-1}\left(\int p(x)dx\right)\left[C + \int q(x)\cdot\exp\left(\int p(x)dx\right)dx\right]$$

```
import sympy as sm
from sympy import symbols
from sympy.parsing.sympy_parser import parse_expr

# Definicion de variables simbolicas
x,C = symbols('x C')
p = '1/(10-x)'
q = '-100/(10-x)'

# Solucion de la ED lineal de primer orden
phi = sm.exp(sm.integrate(p,x))
solution = C/phi+sm.integrate(phi*parse_expr(q),x)/phi
```

En la última linea, se debe operar $q(x)$ que es una de caracteres con una variable simbólica $\phi(x)$. Si no se tiene en cuenta que es necesario hacer una conversión de datos, la ejecución del programa saca un error debido a que no se pueden operar cadenas de caracteres con elementos simbólicos de scipy. Algo parecido ocurre cuando se intenta operar una cadena de caracteres o un caracter con un valor numérico (ej. '2'+3*4).

2 Módulos

El gran universo de Python puede ser agrupado a través de familias de módulos. Estos módulos son usados en muchas áreas del conocimiento para facilitar la interpretación y procesamiento de problemas complejos, optimizar tareas, entre otras ventajas. Estas familias tienen la finalidad de ofrecer herramientas para cumplir un objetivo específico. Por ejemplo:

- Análisis numérico: numpy, scipy
- Visualización: matplotlib, bokeh, plotly
- Modelos: tensorflow, sklearn
- Manejo de datos estructurados: pandas, dask
- Interfaz gráfica: tkinter, dash, ipywidgets
- Procesamiento digital de imágenes: cv2, pillow

2.1 Instalación

Para instalar módulos en Python, se puede hacer de tres formas:

- Desde Anaconda:

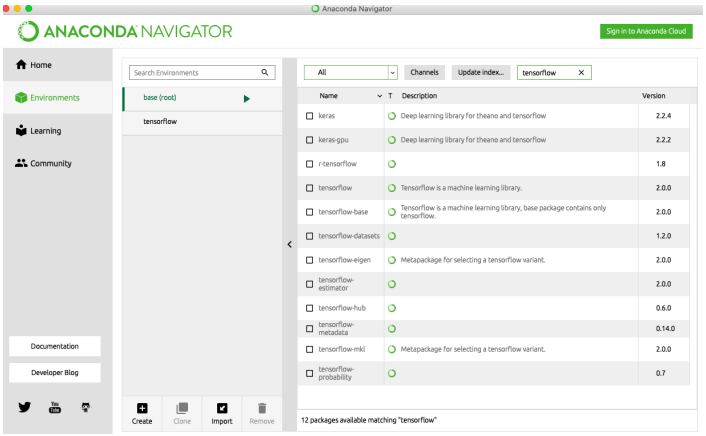


Figura 1: Ejemplo de creación, manipulación y eliminación de variables

- Desde Anaconda Prompt (Windows) ó Consola (Linux, Mac):

```
pip install modulo_1,modulo_2,...,modulo_n
```

- Desde un código:

```
!pip install modulo_1,modulo_2,...,modulo_n
```

El siguiente comando permite listar los módulos instalados en un dispositivo.

```
pip list
```

2.2 Uso

Existen varias formas de utilizar los módulos en Python, esto se define en la importación.

- Importando el módulo con el nombre que está definido por defecto. Por ejemplo:

```
import os
files = os.listdir()
```

- Importando el módulo y renombrado el nombre que está definido por defecto con un alias. Por ejemplo:

```
import numpy as np
import pandas as pd
x = np.array(10)
df = pd.DataFrame(x,columns = ['var_x'])
```

- Importando la función que se necesita del módulo. Esto es útil cuando se requieren usar una función específica de uno o varios módulos.

```
from ipywidgets import widgets, interactive
from ipywidgets import interactive_output as inter_out
from IPython.display import display

wid_slider = widgets.Slider(
            min = 0,
            max = 10,
            step =1,
            value = 1,
            description='Slider: '
            )
def update(value):
  print(value)

output = inter_out(update,{'value':wid_slider})
display(outpuy)
```

3 Documentación y buenas prácticas de programación

Cuando se escribe código, es importante tener ciertas pautas y normas para que otros programadores puedan entender y replicar nuestro trabajo. Es importante que todo el código sea escrito en un mismo lenguaje y es recomendado usar inglés. Para estas prácticas, se puede optar por usar inglés o español.

3.1 Definición de variables

Se debe definir un estándar para nombrar las variables de acuerdo a uno de los siguientes posibles casos:

- Variables definidas en minúscula separadas por underscore. (ej. str_year, int_count_cycle, df_vehicle)
- Se definen en minúscula con la primer letra de cada palabra en mayúscula. Esto se conoce comoUpperCase. (ej. StrYear, IntCountCycle, DfVehicle)

Para estas prácticas, se debe de usar el primer caso. Las variables se deben definir según la siguiente nomenclatura: **type_name** donde type representa el tipo de dato y name la descripción de la variable. Es importante ser explícitos al nombrar la variable de modo que el nombre sea coherente con el uso que tiene en el código.

3.2 Definición de funciones

Las funciones se deben nombrar con el estándar definido para las variables. Por ejemplo: si se desea crear una función que reciba un número y retorne el factorial de éste, el código con el estándar debe ser:

```
def calculate_factorial_number(int_n):
  int_result = 1
  for int_count in range(1,int_n+1):
    int_result = int_result*int_count
  return(int_result)

int_n = 3
int_fact_n = calculate_factorial_number(int_n)
print('El factorial de {} es
    {}'.format(int_n,int_fact_n)
```

3.3 Comentarios

Los comentarios permiten documentar el flujo de un código y describir funciones y algunas instrucciones claves Se hacen antes de la declaración de la sentencia a evaluar o de la función. Existen tres tipos de comentarios:

- **Por bloque:** Se usan principalmente para hacer seguimiento de funciones y métodos. Por lo general, estos comentarios van con un encabezado.

```
''' -- Declaracion funcion para calcular
    factorial --

Esta funcion calcula el factorial de un numero a
    traves de un ciclo for

- Variables entrada: int_n
- Variables de salida: int_result
'''
def calculate_factorial_number(int_n):
  int_result = 1

  for int_count in range(1,int_n+1):
    int_result = int_result*int_count
  return(int_result)

int_n = 3
int_fact_n = calculate_factorial_number(int_n)
print('El factorial de {} es
    {}'.format(int_n,int_fact_n)
```

- **Descriptivos:** Son un complemento del comentario por bloques y van en lugares donde se considere necesario dar claridad en alguna etapa del código. Estos comentarios se hacen con

```
def calculate_factorial_number(int_n):
  int_result = 1
  # Se usa el ciclo for para multiplicar
      int_result con el valor int_count que
      varia 1 a 1 hasta llevar a int_n+1
  for int_count in range(1,int_n+1):
    int_result = int_result*int_count
  return(int_result)

# Se definen variables para probar el programa
int_n = 3
# Se hace llamado de la funcion definida
    previamente
int_fact_n = calculate_factorial_number(int_n)
print('El factorial de {} es
    {}'.format(int_n,int_fact_n)
```

- **Debugging:** Son comentarios de una linea y se hace para hacer debugging y seguimiento del algoritmo.

4 Pautas de entrega y evaluación

La evaluación relacionada con Python se hará a través de tres talleres cada uno independiente del otro. El taller de manejo de datos y el taller de manejo de sentencias implica el desarrollo de todos los ejercicios. Para el taller de manejo de módulos y funciones, cada grupo deberá seleccionar uno de los problemas planteados y darle solución a través de la definición de funciones (sub-programas) y los módulos de Python involucrados en el problema.

El porcentaje de evaluación de cada taller es:

- Práctica de manejo de datos (5%)
- Práctica de sentencias (5%)
- Práctica de módulos y funciones (10%)

Las grupos deben tener mínimo dos miembros y máximo tres. El archivo que se debe entregar en cada fecha de entrega es un archivo comprimido (.zip) que contenga la información presentada en la figura 2. En todas las prácticas, el 10% de la evaluación la conforman el tema de nomenclatura y definición de variables de acuerdo al estándar presentado previamente, comentar el código y explicar brevemente en los cuadernos de Jupyter aspectos claves del algoritmo.

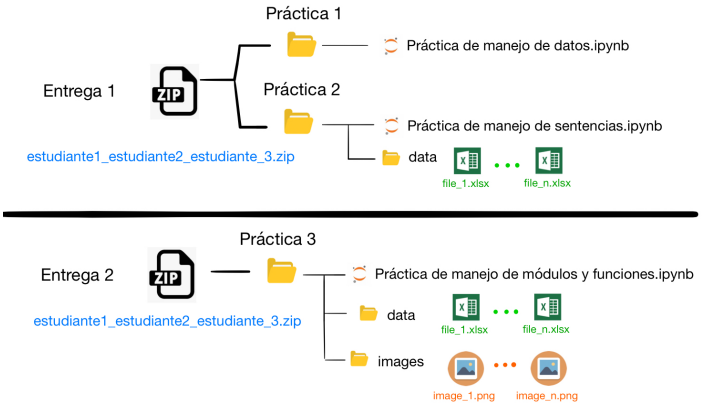


Figura 2: Estructura de la entrega

Las entregas se deben realizar según las siguientes fechas:

- Práctica de manejo de datos y práctica de manejo de sentencias. **18 de Mayo hasta las 12 de la noche**
- Práctica de manejo de módulos y funciones. **29 de Mayo hasta las 12 de la noche**

El desarrollo de las prácticas está sometido a sustentación. Las fechas se asignarán de acuerdo a los grupos conformados en el Google Drive.

Anexo a este archivo, se compartió un link con una hoja de excel de Google Drive en la cual deben definir los grupos de trabajo a más tardar el 05 de Mayo.

5 Práctica de manejo de datos – 5%

La caracterización y regulación de los vehículos que circulan en una región están determinados por una serie de pruebas que buscan diagnosticar los sistemas mecánicos y de emisiones de los vehículos. Una prueba muy común en los CDA es una prueba de evaluación de emisiones de humo generadas por las fuentes móviles accionadas con diésel a través de un método de aceleración libre. Esta prueba está especificada en la norma NTC 4231.

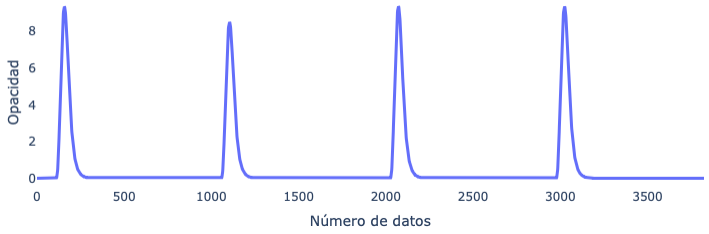


Figura 3: Perfil de opacidad para una prueba NTC 4231

La norma consiste en realizar cuatro ciclos de aceleración libre con el vehículo en ralentí (el motor encendido pero el vehículo no se mueve, velocidad y aceleración nulas). Cada ciclo se hace de la siguiente manera: Se acelera súbitamente por dos segundos hasta que el motor alcance una velocidad angular específica (rpm gobernada). Una vez se alcanza esta velocidad, se libera el acelerador para que el motor regrese a velocidad de ralentí. Existen otras consideraciones técnicas que se pueden encontrar en la norma.

El primer ciclo se realiza para que el operario se familiarice con la secuencia y para eliminar el hollín que se puede tener en el sistema de escape del vehículo. Los tres ciclos posteriores se realizan en intervalos de 20 segundos. Como se resultado, se obtiene un perfil de opacidad como el mostrado en la figura 3.

La información que se obtiene durante la prueba se especifica en la siguiente tabla. También se obtiene la temperatura del aceite del motor antes y después de la prueba. Es común que toda estos resultados y la información general del vehículo se registren y posteriormente se almacenen en archivos pdf.

Tabla 1: Variables y asignación de nomenclatura para la práctica.

Ciclo	RPM	Opacidad
1	rpm_1	opacity_1
2	rpm_2	opacity_2
3	rpm_3	opacity_3
4	rpm_4	opacity_4
	rpm_prom	opacity_prom

RPM gobernada	Diámetro del tubo de escape	Opacidad nominal
rpm_gob	d_exhaust	opacity_ref

Se debe realizar un programa que extraiga información de los registros de cientos de vehículos. Una forma de extraer estos datos es usando procesamiento de texto.PyPDF2 es uno de varios módulos de extracción de texto de archivos pdf.

```
import PyPDF2

def extract_one_page(pdf_name):
  read_pdf=PyPDF2.PdfFileReader(pdf_name)
  page=read_pdf.getPage(0)
  page_content=page.extractText()
  return(page_content)
```

En el anterior código, se define una función (un sub-programa) que recibe el nombre de un archivo pdf y retorna como cadena de caracteres todo el contenido de la página 0 del archivo. Este script se incorporó en un programa más complejo y como resultado, se obtuvo para una amplia gama de vehículos una cadena de caracteres como la siguiente:

> 'RPM Gobernada Medida [1/Min]2820RPM Ralentí Medida [1/Min]604Temp. Motor [ºC]75Opa. Ciclo1 [%]3,3RPM C1 [1/Min.]2820Opa. Ciclo2 [%]2,7RPM C2 [1/Min.]2810Opa. C3 [%]2,8RPM Ciclo3 [1/Min.]2820Opa. C4 [%]2,2RPM Ciclo4 [1/Min]2820Opa. Prom. [%]2,57Valor Norma[%]35LTOE [mm] (Diámetro tubo de escape)65Temp. Fin Mot. [ºC]74RESULTA'

Estas cadenas tienen la misma estructura y tienen 325 elementos. Con esta guía, se adjuntó un cuaderno de Jupyter con nombre "Práctica de manejo de datos.ipynb" que contiene las cadenas de caracteres obtenidas para cinco vehículos.

5.1 Evaluación

- **PARTE I – 40%** Se debe desarrollar un programa que extraiga los datos numéricos y las variables de cada cadena de caracteres y los guarde en un dataframe. Las columnas de este dataframe son las presentadas en la tabla 1 incluyendo la temperatura del aceite del motor antes y después de realizar la prueba. Utilice métodos de cadenas de caracteres para extraer los datos y utilice pandas para guardarlos. Posteriormente, ejecute la siguiente función que anexa al dataframe una columna llamada "test_result" que hace indica si la prueba se rechaza o no.

```
df['test_result'] = return_test_result(df)
df.head()
```

- **PARTE II – 40%** Se debe de desarrollar un programa que pregunte al usuario un vehículo para consultar, el programa debe de mostrar los resultados de la prueba para el vehículo seleccionando. Los resultados que se deben mostrar son:
  - La designación del vehículo
  - RPM promedio
  - Opacidad promedio
  - El resultado de la prueba
- **PARTE III – 10%** Utilice la función suministrada en el cuaderno para realizar una gráfica de la opacidad en función de las revoluciones del motor. El llamado se debe hacer de la siguiente manera:

```
fig = plot_summary_test(df)
fig.show()
# Ayuda: df es un dataframe de 5 filas por 16
    columnas. Las columnas deben de tener los
    nombres de la tabla.
```

- **PARTE IV – 10%** Documentación y explicación de los algoritmos. Esto no se hace como una sección a parte sino que se debe de incorporar en las otras tres etapas de la práctica. (Ver guía de documentación)

5.2 Recomendaciones

- Antes de empezar el desarrollo de este trabajo, se recomienda que instale los módulos de cada cuaderno. En la guía hay una sección donde se presentan comandos donde se muestra con más detalle como realizar estas instalaciones.
- No se puede extraer los datos accediendo a los indices de la cadena de caracteres, es decir, extraer variables como vehicle_1[26:29]. Utilice los comandos replace() y split() para la extracción de los datos.
- El tema de definición de variables y documentación también hacen parte de la evaluación de esta práctica.

6 Práctica de manejo de sentencias – 5%

La dinámica es el estudio de las fuerzas y cómo estas alteran el movimiento de un cuerpo. Para un vehículo, la dinámica se define a través de la potencia específica vehicular (*Vehicle Specific Power – VSP*). Esta variable se define como la potencia que un vehículo posee para estar unas condiciones dinámicas y topográficas dadas. Además, es la contribución de diferentes aportes energéticos que posee un vehículo durante un recorrido y matemáticamente es:

$$VSP = \frac{1}{m}\left(\frac{d}{dt}(E_k + E_p) + v \cdot F_{roll} + v \cdot F_{aero}\right) \quad (1)$$

Luego de evaluar las derivas y reemplazar las definiciones de fuerzas y energías en la anterior ecuación se llega a:

$$VSP = v \cdot (a \cdot (1+\varepsilon) + g \cdot \mathbf{grade} + g \cdot C_R) + \frac{1}{2 \cdot m}\rho_a \cdot C_D \cdot A \cdot v^3 \quad (2)$$

La velocidad del vehículo v, la aceleración del vehículo a, y la pendiente de la vía del recorrido **grade** son perfiles temporales. La masa equivalente de componentes rotativos ε, el coeficiente de rodadura C_R, el coeficiente aerodinámico C_D, el área frontal A y la masa del vehículo m son parámetros del vehículo y comúnmente se asumen constantes durante los recorridos. ρ_a es la densidad del aire. Esta variable es de utilidad a la hora de explicar los efectos dinámicos de un vehículo durante un recorrido determinado, estimar consumo de combustible y estimar emisiones contaminantes. Además, permite deducir otras variables dinámicas más complejas.

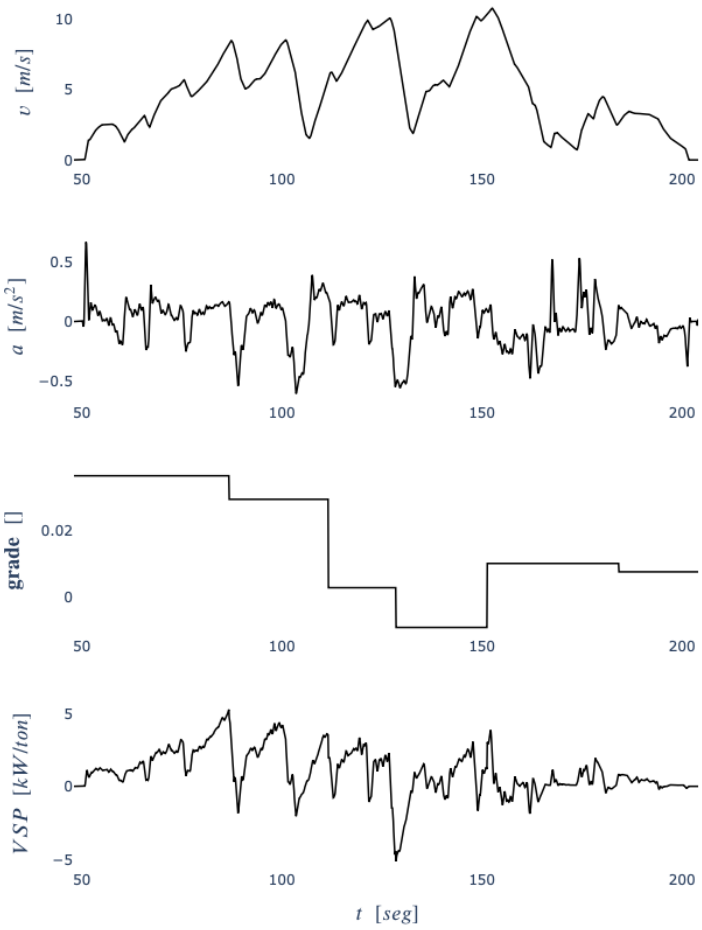


Figura 4: Perfiles dinámicos de una prueba

En el cuaderno anexo a la práctica se encuentra una carpeta con nombre **"data"** que posee resultados para cinco pruebas. En cada uno de estos archivos se encuentran datos de tiempo, velocidad, aceleración y pendiente obtenidos con una frecuencia de 10 Hz. En la figura 4 se presenta un perfil temporal para estas variables.

En los sistemas de medición hay dos términos importantes que se relacionan y son completamente diferentes. El primero es el tiempo de muestreo, el cual representa el tiempo segundo a segundo que transcurre durante una prueba. El segundo es la frecuencia de adquisición de los datos o frecuencia de muestreo. Esta representa la cantidad de datos que se guardan en un segundo. Para una ventana de t segundos, la cantidad de datos que se debe de tomar es $n = f \cdot t$

6.1 Evaluación

- **PARTE I – 50%** Utilice el comando listdir() del módulo os para obtener los archivos contenidos en la carpeta "data" como una lista de cadena de caracteres. Posteriormente, realice un ciclo iterativo que cargue uno a uno estos archivos. Para esto utilice pandas. En cada iteración el programa debe:
  - Calcular la VSP de acuerdo a la ecuación 2. Asuma $\varepsilon = 0.42$, $C_R = 6.5 \times 10^{-3}$, $C_D = 0.75$, $\rho_a = 1\ kg \cdot m^{-3}$, $A = 7\ m^2$, $m = 9.6\ ton$, $g = 9.81 m \cdot s^{-2}$. Esta variable se debe anexar al dataframe de cada prueba.
  - Realizar un ciclo iterativo que inicia en 3, va hasta 30 con un paso de 3. Esta variable representa un tiempo para estimar un perfil de VSP promedio. Por ejemplo, si el ciclo está evaluando 9, esto indica que el perfil de VSP promedio se hará con una ventana de 9 segundos hacia atrás. Esta evaluación se hace dato a dato en todo el eje temporal. Estos perfiles se deben guardar en el dataframe de cada prueba.
  - Almacenar cada dataframe en un diccionario para su posterior uso.
- **PARTE II – 40%** Realice un gráfico comparativo para cada una de las cinco pruebas en el cual presente los perfiles de la VSP promedio vs tiempo. Este gráfico debe ser parecido al presentado en la figura 5. Para esto, realice un ciclo for para recorrer el diccionario almacenado en la etapa previa.

Figura 5: Perfiles dinámicos de una prueba

- **PARTE III – 10%** Nomenclatura de variables, documentación y explicación de los algoritmos. Esto no se hace como una sección a parte sino que se debe de incorporar en las otras tres etapas de la práctica. (Ver guía de documentación)

6.2 Recomendaciones

- Antes de empezar el desarrollo de este trabajo, se recomienda que instale los módulos de cada cuaderno.
- En caso de tener errores de lógica de programación realice una depuración de lo que hace internamente su algoritmo. Para esto, puede usar comandos print durante la ejecución del algoritmo, como:

```
for i in range(10):
  print(i)
  # Se anexa el contenido del ciclo for. Se
      espera que calcule algo de acuerdo al
      valor que toma i
  print(i<5)
```

7 Práctica de manejo de módulos y funciones – 10%

En esta práctica, el equipo debe seleccionar uno de los problemas planteados como ejercicios aplicativos. Para esto deben usar el módulo "ipywidgets" y los módulos relacionados con cada problema.

7.1 Evaluación

- **PARTE I – 50%** Reproducibilidad de la aplicación. Esto implica que la aplicación debe cumplir con todos los requerimientos de funcionalidad solicitados.
- **PARTE II – 10%** Consulta sobre los módulos y funciones usados en la práctica. Esto se debe hacer en el mismo cuaderno.
- **PARTE III – 15%** Un diagrama explicando cómo se subdivide el problema que se plantea y su equivalente diagrama de flujo.
- **PARTE III – 15%** Prueba de escritorio para al menos 6 casos diferentes. En el cuaderno está la sección donde se deben de realizar estas pruebas de escritorio.
- **PARTE IV – 10%** Nomenclatura de variables, documentación y explicación de los algoritmos. Esto no se hace como una sección a parte sino que se debe de incorporar en las otras tres etapas de la práctica. (Ver guía de documentación).

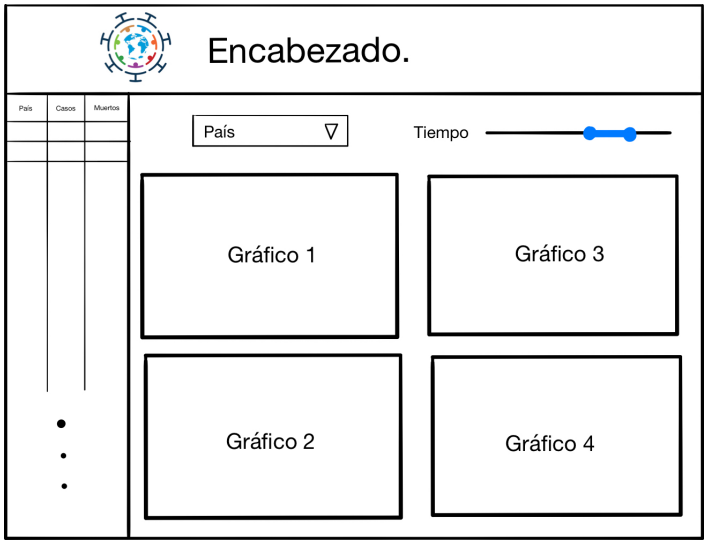
7.2 Recomendaciones

- Antes de empezar el desarrollo de este trabajo, se recomienda que instale los módulos de cada cuaderno. En la guía hay una etapa de comandos donde se muestra con más detalle como realizar estas instalaciones.
- Es obligatorio usar funciones (sub-programas)

7.3 Propuesta de aplicaciones

7.3.1 APT_1 – Interfaz Covid-19

Se requiere hacer una aplicación que permite mostrar un histórico de los casos y muertes asociadas con la pandemia Covid-19. La estructura de la aplicación debe ser parecida a la presentada en la figura 7.3.1.

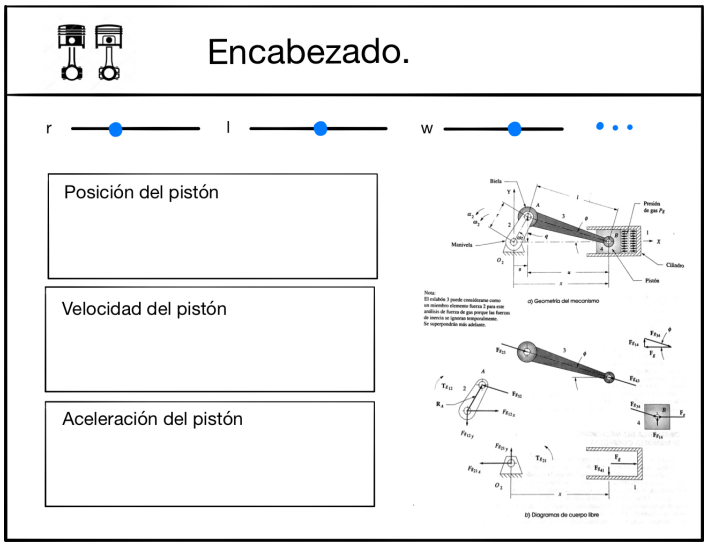


La aplicación debe de cumplir con los siguientes requerimientos:

- La aplicación debe tener un encabezado donde se ponga una imagen descriptiva de la finalidad del aplicativo, el título y los autores.
- Se debe ubicar una tabla ordenada por países de mayor a menor contagios. Las variables que se deben mostrar en esta tabla son el país, la cantidad de casos y la cantidad de muertos.
- Se debe mostrar cuatro figuras que varían dependiendo del país y el rango de tiempo seleccionado. Estas figuras deben mostrar el siguiente contenido:
  - **Up - Left** Un perfil temporal de la cantidad de casos y un perfil de cantidad de casos acumulados.
  - **Up - Right** Un perfil temporal de la cantidad de muertos y un perfil de cantidad de muertos acumulados.
  - **Down - Left** La derivada temporal de la cantidad de casos.
  - **Down - Right** La derivada temporal de la cantidad de muertos.
- Los widgets que se definan para seleccionar el país y el rango temporal deben estar ordenados de menor a mayor.

7.3.2 APT_2 – Interfaz dinámica del mecanismo biela manivela corredera

El mecanismo biela manivela corredera es muy común en muchas maquinas incluyendo los motores de combustión. El aplicativo que debe desarrollar es un simulador dinámico que permita visualizar las variables cinemáticas del mecanismo variando algunos parámetros. La estructura de la aplicación debe ser parecida a la presentada en la figura.



Sea r el radio de la manivela y l la longitud de la biela. El ángulo de la manivela es θ y el ángulo que forma la biela con el eje X es ϕ. Para cualquier velocidad angular constante de la manivela ω, el ángulo de ésta es $\theta = \omega t$. La posición instantánea del pistón x. Las ecuaciones cinemáticas del pistón son:

$$x \cong l - \frac{r^2}{4l} + r\left(\cos(\omega t) + \frac{r}{4l}\cos(2\omega t)\right) \tag{3}$$

$$\dot{x} \cong -r\omega\left(\sin(\omega t) + \frac{r}{2l}\sin(2\omega t)\right) \tag{4}$$

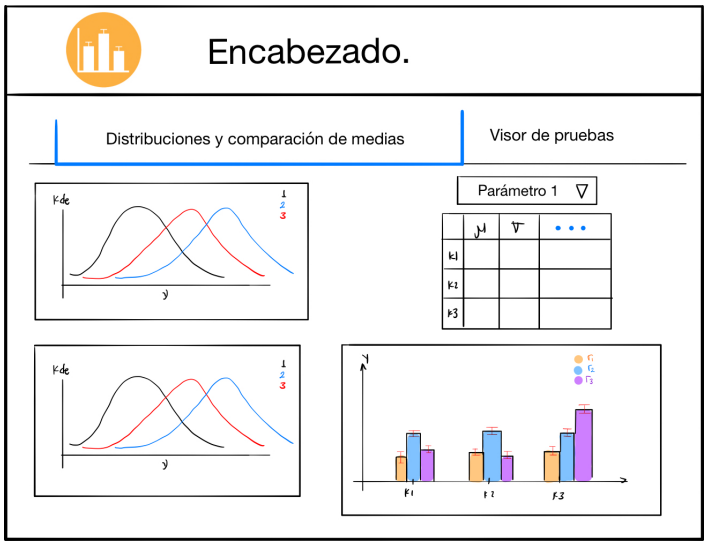
$$\ddot{x} \cong -r\omega^2\left(\cos(\omega t) + \frac{r}{l}\cos(2\omega t)\right) \tag{5}$$

La aplicación debe de cumplir con los siguientes requerimientos:

- La aplicación debe tener un encabezado donde se ponga una imagen descriptiva de la finalidad del aplicativo, el título y los autores.
- La aplicación debe permitir seleccionar l, r, ω y t_{max} y mostrar las variables cinemáticas en los gráficos.
- El dibujo anexo debe ser mostrado en la aplicación.
- ¿Qué tan factible es incorporar la velocidad angular variable en el tiempo al problema?

7.3.3 APT_3 – Analizador estadístico

La estadística es una ciencia muy utilizada hoy en día. Es la base para llegar a tener algoritmos inteligentes que pueden simular comportamientos humanos. Este aplicativo debe mostrar una análisis estadístico descriptivo para una base de datos que posee un estándar definido. La estructura de la aplicación debe ser parecida a la presentada en la figura.



El estándar debe seguir un diseño experimental donde se evalúan dos parámetros de interés k y r para una variable y. Se usó un diseño experimental como el que se presenta en la tabla 2. Los resultados de estos datos se presentan en la carpeta "apt_3/data/" como archivos independientes para cada variable.

Tabla 2: Diseño experimental

		Parámetro 1		
		k_1	k_2	k_3
Parámetro 2	r_1	y_{11}	y_{12}	y_{13}
	r_2	y_{21}	y_{22}	y_{23}
	r_3	y_{31}	y_{32}	y_{33}

La aplicación debe de cumplir con los siguientes requerimientos:

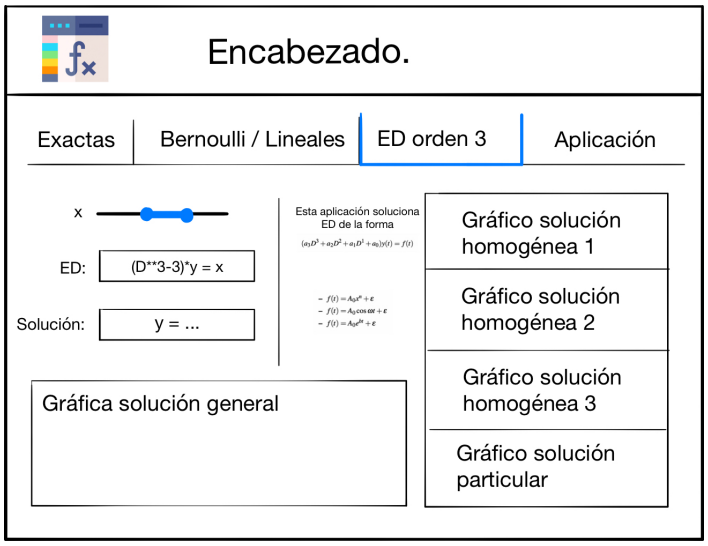
La aplicación debe tener dos pestañas. A continuación se detalla que información se pone en cada una de ellas.

- **Distribuciones y comparación de medias:** Se debe presentar tres gráficos. El primero presenta la distribución para la variable en los posibles casos del parámetro 1. El segundo presenta la distribución para los datos de para los posibles casos del parámetro 2. El eje x representa el valor medido de la variable y el eje y representa la distribución de estos. Utilice leyendas para identificar los casos de cada variable. El tercero es una figura donde se presente el promedio y un intervalo de confianza al 95% de confianza estadística para la variable y en los parámetros evaluados mediante un gráfico de barras. Adicionalmente, se debe presentar una tabla con los parámetros estadísticos (media, mediana, kurtosis, skewness) para un parámetro. El parámetro se selecciona con un widget.
- **Visor:** En esta pestaña, se deben ponder dos widgets que permitan seleccionar un valor de k y un valor de r para presentar en una figura la variable y en función del número de datos.

Este aplicativo, permite hacer un análisis estadístico básico para cualquier diseño experimental que tenga la estructura presentada en la tabla 2. Para esta práctica, se presentan tres bases datos dentro de la carpeta data que permiten evaluar el algoritmo.

7.3.4 APT_4 – Solucionador de ecuaciones diferenciales

Las ecuaciones diferenciales permiten describir el comportamiento de un sistema. Son muchas familias y en muchos casos, muy compleja su solución. Para este aplicativo, se requiere hacer un solucionador de ecuaciones diferenciales para varios grupos de familias. La estructura de la aplicación debe ser parecida a la presentada en la figura.



La aplicación debe de cumplir con los siguientes requerimientos:

El aplicativo debe conformarse por tabs. Cada tab debe contener una interfaz que permita ingresar los términos de la ecuación diferencial y darle solución. Así mismo, se debe presentar gráficamente estos resultados.

- ED exactas o reducibles a exactas.
- ED bernoulli y lineal
- ED lineal de tercer orden de coeficientes constantes

$$(a_3D^3 + a_2D^2 + a_1D^1 + a_0)y(t) = f(t)$$

  La aplicación debe permitir seleccionar un tipo de familia para $f(t)$ y posteriormente preguntar por los parámetros de cada familia:

  - $f(t) = A_0x^n + \varepsilon$
  - $f(t) = A_0\cos\omega t + \varepsilon$
  - $f(t) = A_0e^{bt} + \varepsilon$

- Aplicaciones de las ecuaciones diferenciales. Para esta práctica se debe hacer uno de los siguientes casos.

  - Crecimiento exponencial con límite máximo de población
  - Vaciado de tanques.
  - Mezclas químicas
  - Dinámica de partículas
  - Ley de enfriamiento
  - Circuitos eléctricos