


Desarrollo y Plataformas Utilizadas en las Sociedades Fintech

Objetivos

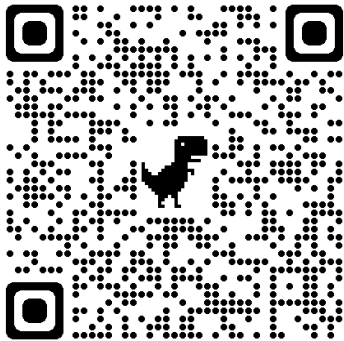
1. Conocer principios básicos de programación en lenguaje Python
2. Conocer las librerías de análisis de datos y graficas
3. Conocer conceptos de Blockchain y otras innovaciones
4. Conocer el ecosistema Fintech en México

Bienvenidos

	LUN 18	MAR 19	MIÉ 20	JUE 21
GMT-06				
10 AM	Inicio - Diagnóstico, 9:30am Introducción Python 10 – 11am	Python 4 9:30 – 11am	Python 7 9:30 – 11am	Fintech 1 9:30 – 11am
11 AM	Receso 15min, 11am Python 2 11:15am – 12:30pm	Receso 15min, 11am Python 5 11:15am – 12:30pm	Receso 15min, 11am Python 8 11:15am – 12:30pm	Receso 15min, 11am Fintech 2 11:15am – 12:30pm
12 PM	Receso 15min, 12:30pm Python 3 12:45 – 2:30pm	Receso 15min, 12:30pm Python 6 12:45 – 2:30pm	Receso 15min, 12:30pm Intro Blockchain 12:45 – 2:30pm	Receso 15min, 12:30pm Evaluación 12:45 – 2pm
1 PM				
2 PM				Cierre, 2pm
3 PM				

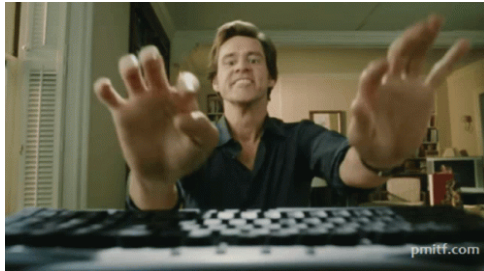
Evaluación diagnóstica

- Por favor, contesta el siguiente cuestionario:
- <https://forms.gle/3W4sNpEcFaSdMkG1A>



Metodología

La mejor forma de aprender programación es: **PROGRAMANDO**



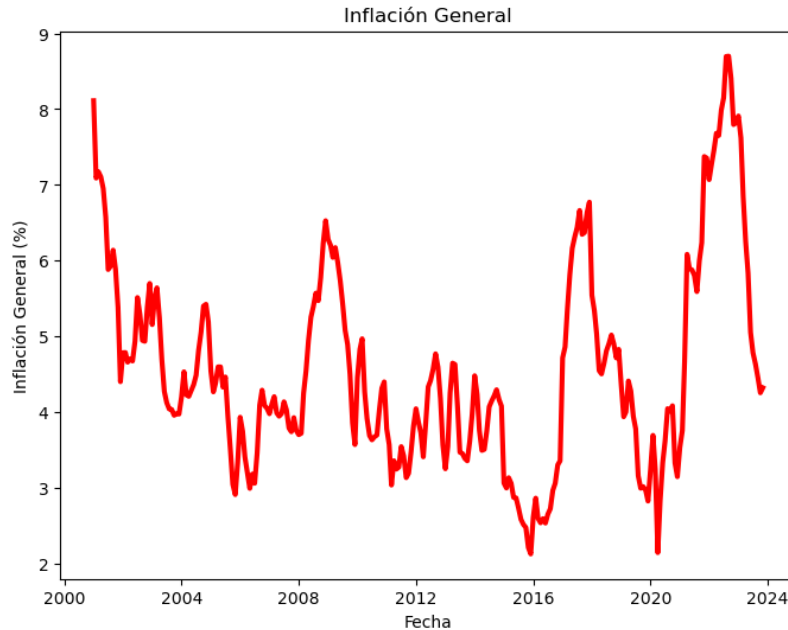
Para las clases de programación Python, utilizaremos un ambiente en la nube.

1. **Cuadernos de ejercicios Jupyter Lab:** Permiten combinar texto y código en tiempo real
2. **Plataforma MyBinder:** Genera un ambiente de programación
3. Cada participante tendrá su propio ambiente de trabajo

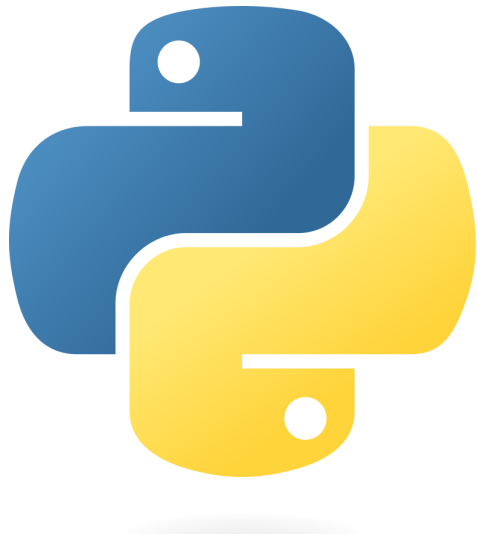
Importante: El ambiente de trabajo se "apaga" después de 10min de inactividad y es posible que la información hasta el momento se pierda. Si desean mantener su trabajo, es necesario que guarden sus archivos de trabajo en forma local.

Al finalizar

- Vamos a poder realizar un reporte consultando información en línea del INEGI y otras bases de datos:



Introducción a Python



Python

Top 10 en Lenguajes

PYPL Popularity of Programming Language Index

Posición	Lenguaje	Participación
1	Python	28.09 %
2	Java	15.81 %
3	JavaScript	8.93 %
4	C/C++	6.8 %
5	C#	6.64 %
6	PHP	4.6 %
7	R	4.53 %
8	TypeScript	2.81 %
9	Swift	2.8 %
10	Objective-C	2.33 %

¿Por qué aprender Python?

- Programas fáciles de leer y comprender debido a su sintaxis básica es similar a la del inglés.
- Incrementa la productividad de los programadores, con menos líneas de código en comparación con muchos otros lenguajes.
- Cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python con otros lenguajes de programación conocidos, como Java, C y C++.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo.
- Python se puede portar a diferentes sistemas operativos como Windows, macOS, Linux y Unix.

Python es un lenguaje de alto nivel de programación interpretado que hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo.

- Lenguaje de propósito general: puedes construir cualquier cosa!
- Apto para las actividades diarias permite la construcción de prototipos en poco tiempo.
- Open source == Gratis!
- Paquetes disponibles para una gran cantidad de aplicaciones y campos
- Multiplataforma
- Permite múltiples estilos de programación: orientada a objetos, imperativa y funcional
- Primera versión: 20 de febrero de 1991, creado Guido van Rossum



TOP

Los programas en Python más famosos:

1. Pinterest.
2. Panda 3D.
3. Dropbox.
4. Spotify.
5. Netflix.
6. Uber.
7. Instagram.
8. Reddit.
9. Battlefield 2.

Aplicaciones de Python:

1. Análisis de Big Data.
2. Data Science
3. Inteligencia Artificial y Aprendizaje Automático
4. Desarrollo Web.
5. Juegos y Gráficos 3D.
6. Procesamiento de Imágenes y Audio
7. Automatización de procesos

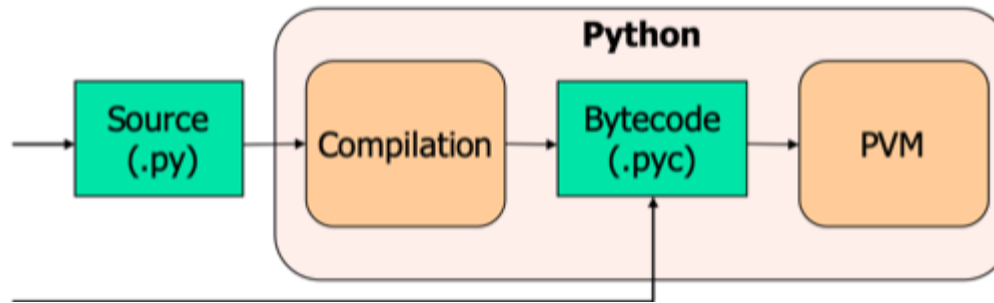
Tipos de lenguajes de programación

- **Compilados:** Se obtiene un archivo ejecutable (.EXE), le damos doble click o ejecutamos la tarea y comienza con la ejecución de las instrucciones
- **Interpretados:** tiene un proceso mucho más sencillo, el intérprete lo que hace es leer el archivo y hacer la traducción(interpretar) lo que entiende, normalmente lo hace línea por línea.
- **Lenguaje ensamblador:** Es un código a nivel procesador que se ejecutan directamente por lo que no es necesario compilarlos o interpretarlos.

	Compilado	Interpretado
Características	C++	Python
Ventajas	Rapidez Listo para ejecutarse	Flexibilidad Multiplataforma
Desventajas	El ejecutable corre en una sólo plataforma Para realizar cambios, hay que recompilar	Más lento Se debe interpretar el código en cada ejecución
Ejemplos	C++, Rust, GO	Javascript, Python, Ruby

Modelo de ejecución

- Python compila el código fuente a bytecodes
 - Almacenados como ficheros .pyc para reutilización
- Python Virtual Machine (PVM): interpreta los bytecodes



Ejemplos de un programa básico en varios lenguajes

- BASIC

```
10 PRINT "Hello, World!"  
20 END
```

- C++

```
main( ) {  
    printf("hello, world")  
}
```

- Python

In []:

Instalación de Python

Solo hay que descargarlo en el formato adecuado para nuestro sistema operativo desde <https://www.python.org>

En muchos sistemas Unix/Linux, Python viene ya instalado por defecto (incluso es usado por el propio S.O.).

NOTA: Si la instalación es en Windows marcar la opción de añadir Python al PATH. Ver notas en: <https://docs.python.org/3/using/windows.html>

También puede venir incluido en distribuciones de software más amplias:

- El ejemplo más conocido es **Anaconda**, que incluye numerosos paquetes científico/técnicos de análisis de datos adicionales (así como un IDE, una GUI, un gestor de paquetes/entornos (**conda**), etc.)
 - Existe una versión minimalista llamada **Miniconda**, que solo incluye Python, *conda*, y unos pocos paquetes extra.

Instalación de paquetes externos

La herramienta oficial (y más usada) para instalar paquetes Python es `pip`.

Instala paquetes del Python Package Index (PyPi): <https://pypi.org/>

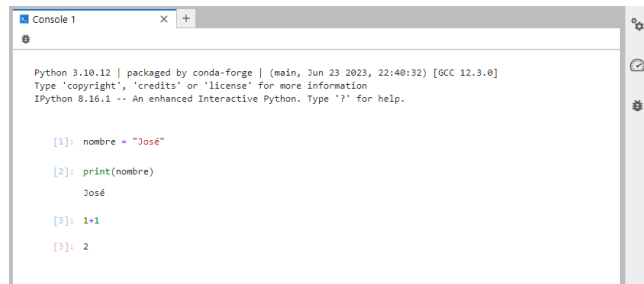
```
pip install matplotlib
```

Revisaremos la instalación y utilización de librerías en otra sesión

Ejecución de Python

Interpretes interactivos

- Intérprete interactivo (consola):



```
Python 3.10.12 | packaged by conda-forge | (main, Jun 23 2023, 22:40:32) [GCC 12.3.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.16.1 -- An enhanced Interactive Python. Type '?' for help.

[1]: nombre = "José"

[2]: print(nombre)
     José

[3]: 1+1

[3]: 2
```

- las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente
- da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.
- útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

Nota: En el curso trabajaremos en los cuadernos de trabajo, pero pueden explorar la consola.

In []:

En *scripts*

```
python myscript.py
```

O ejecutándolo:

```
myscript.py
```

- En Unix, se usa algo como `#!/usr/bin/env python`, en la primera línea del script.
- En Windows, se puede hacer *doble click* con el ratón (cuidado: se puede cerrar sin poder ver el resultado)

Los programas suelen necesitar acceder a librerías externas (en Python, módulos), otros ficheros `.py`.

- Volveremos sobre ello, pero los módulos se buscan o bien el directorio actual, o en algunos directorios del sistema, o en los indicados por la variable de entorno `PYTHONPATH`.

Edición de scripts y módulos

Se puede usar cualquier editor de texto puro: `vim`, `emacs`, `sublime`, `notepad++` ...

- No usar algo como `Word` !

También se puede usar un entorno de desarrollo integrado (IDE)

- Incluyen muchas utilidades de ayuda, depuración, refactorización
- Integran un intérprete interactivo
- Muchas opciones: IDLE (parte de Python), PyCharm, Spyder (de Anaconda), Visual Studio, Eclipse...

O se puede usar *Jupyter*.

El código fuente en Python

La indentación tiene significado sintáctico en Python

- Python anima a escribir código legible
- La indentación (espacios o tabuladores) crea bloques de código
- Se recomienda usar siempre 4 espacios (y evitar el tabulador)

Lo mismo en Python:

```
if X > 0:  
    print("X es positivo")  
elif X < 2:  
    print("X es 1")
```

En Python, los comentarios se introducen con #:

```
# This is a comment  
print("Hello")
```

Jupyter

Jupyter notebooks

- Son documentos que integran código, resultados, visualizaciones y texto en formato *Markdown* (se muestra como HTML).
- Soporta diferentes lenguajes (Python, R, Julia...)
- El usuario interacciona con los notebooks a través de su navegador.
- Muy útil para documentar y compartir trabajo científico
- También muy práctico para pre-configurar entornos de trabajo: ideal para cursos, p. ej.
- También es una opción como entorno de desarrollo/pruebas
- Usando el navegador, podemos ejecutar el código del notebook en el ordenador local, o en un sistema remoto (p.ej. en la nube)

Esquema de trabajo en el curso

