Escuela de Código para PILARES

Descripción de actividades

Parte 3: Programación(MP)

Créditos[1]

Redacción de actividades
Alejandra Sarahí Monroy Velázquez, Carla Irena Blenda Palacios, Karen Alexa Alva Aguirre y Karina Flores García

Coordinación de módulo
Victor Manuel Lomas Barrie

Coordinación de la Transversalización de la Perspectiva de Género
Yuliana Ivette López Rodríguez

Revisiones
Karen Itzel Bruno Sainos, Citlalli Sánchez Mendoza, Carmen Daniela Garrido Juvencio

Supervisión PILARES
Jesús Alanis Manriquez, René Alejandro Rivas Robles y María del Rocío Estrada Monroy

Supervisión IIMAS
Adrián Durán Chavesti, Andrea García Ruiz, Elisa Mariana Valdés Armada, Héctor Alfonso Islas García, Héctor Benítez Pérez, Helena Gómez Adorno, Ivan Vladimir Meza Ruiz, Luz Elena Rueda Rojas, Nora Isabel Pérez Quesadas, María del Pilar Ángeles, Zian Fanti Gutierréz

Financiamiento:
Diseño de un programa de estudios para la capacitación en programación y habilidades en tecnologías de información y comunicación para la escuela de código dentro de PILARES de la Ciudad de México (SECTEI/284/2019)

[1] En orden alfabético.

Agradecimientos

Agradecemos el tiempo y la retroalimentación hecha a los materiales a:

- Ante Salcedo González, ITAM - Instituto Tecnológico Autónomo de México
- Blanca Esther Carvajal-Gámez, ESCOM - Escuela Superior de Cómputo - IPN
- Dagoberto Pulido Arias, IPN - Instituto Politécnico Nacional
- Eréndira Itzel García Islas, UNAM - Facultad de Ciencias
- Marco Antonio Moreno Ibarra, CIC - Centro de Investigación en Computación del IPN
- Ricardo Marcelín Jiménez, UAM-I Universidad Autónoma Metropolitana – Iztapalapa
- Salvador Elias Venegas Andraca, ITESM - Instituto Tecnológico y de Estudios Superiores de Monterrey

También agradecemos el apoyo y seguimiento al personal de SECTEI, en particular de:

- José Bernardo Rosas Fernandez
- Federico Antonio Hernández Loranca
- Rogelio Artemio Morales Martínez
- Adrían Eleazar Contréras Martínez
- Benigno Antonio González Núñez

Índice

Taller 1: Aprende a programar jugando Ajedrez

En honor a las organizaciones lideradas por mujeres que en el campo de las Ciencia, Tecnología, Ingeniería y Matemáticas (por sus siglas en inglés STEM) han formado trabajo en equipo, mediante alianzas y acuerdos con el fin de que las nuevas generaciones sean parte de la transformación en estos ámbitos (ver tabla en el documento *Detalles* para conocer el listado de las organizaciones).

Competencia del taller: Programar un juego de ajedrez en *Python* que haga uso de diferentes tipos de variables, arreglos, y listas, con controles de secuencia como son el if, else, while y operaciones con número y caracteres.

Actitudes: Curiosidad, disposición, constancia, persistencia, apertura a la incorporación de nuevos aprendizajes, capacidad de análisis, apertura al diálogo, escucha, trabajo en equipo, intercambio de opiniones, participación activa, interés y apertura a incorporar en las actividades la Perspectiva de Género para el logro de un bien común.

Actividad 0: Git

<table>
<tr><td colspan="2">Aprendizaje esperado: Hacer uso de un sistema de versiones Git en línea para almacenar todo el código y evidencias producidas.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Cuenta en GitHub.
● Cuenta en GitLab.
● Hojas de papel o fichas de trabajo.
● Lápices, plumas o plumones.</td><td>Evidencia:
1. Repositorio en GitHub y GitLab con el nombre “paginaTaller1” que contenga los archivos index.html de la actividad 4 del taller 1 y el archivo Index.css de la actividad 6 del taller 1, ambos del módulo de Programación Web.
2. 11 fichas de consulta (mínimo).</td><td>Retroalimentación:
● Las tarjetas de consulta deberán responder a cada una de las preguntas de la primera parte de esta actividad.
● Deberán tener escrita la pregunta por el anverso y en el reverso, la respuesta.
● El historial del repositorio “paginaTaller1” debe mostrar 2 commits.
● Debe hacerse un commit y push por cada archivo con el que la participante cuente. Es decir, la cantidad de archivos debe ser equivalente a la cantidad de veces que se ejecuté un commit y un push sobre el repositorio.
● Corroborar que no hubo errores al subir los archivos a los repositorios en GitHub y GitLab.
● El repositorio en GitHub y GitLab, debe tener la misma cantidad de archivos y commits.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Desarrollo de la actividad:

Primera parte: Conociendo Git.
Sugerencia de tiempo invertido: 2 horas.

La participante tendrá un acercamiento mucho más completo al sistema de control de versiones Git.

1. Se le pedirá que realice una búsqueda de información mediante algunos enlaces que se le proporcionen, se recomienda utilizar los enlaces anexos al final de este apartado. Tal información será correspondiente a las interrogantes:
    1. ¿Qué es un SVC (Sistema de Control de Versiones)?
    2. ¿Qué es git?
    3. ¿Qué sistemas Git gratis existen?
    4. ¿Para qué sirve el comando *git*?
    5. ¿Para qué sirve el comando *git init*?
    6. ¿Para qué sirve el comando *git add*?
    7. ¿Para qué sirve el comando *git commit*?
    8. ¿Para qué sirve el comando *git push*?
    9. ¿Cuáles son los pasos a seguir para subir un proyecto a un repositorio?
    10. ¿Por qué git permite trabajar colaborativamente en un proyecto?
    11. ¿Qué errores o conflictos pueden surgir al trabajar colaborativamente en un proyecto con git?
2. La respuesta de cada pregunta será redactada en una ficha bibliográfica.

Segunda parte: Usando Git.
Sugerencia de tiempo invertido: 1 hora.

1. Se solicitará a la participante crear una cuenta en GitLab.
2. La participante editará su perfil. Agregará una fotografía y una descripción de su personalidad, por ejemplo: “Participante en proyectos culturales que benefician mi entorno”, “Aprendiz en la Escuela de Código de PILARES”.
3. Se le indicará crear un repositorio local con el nombre “juegoDeAlianzas_Ajedrez”, en el cual ubicará los archivos *index.html* e *Index.css* generados en las actividades 4 y 6 respectivamente, del taller 1 de este módulo.
4. Posteriormente y a través de la secuencia de comandos investigados en la primera parte de esta actividad, deberá subir dichos archivos a un repositorio remoto en su nueva cuenta de *GitLab*, todo esto a través de la consola de git. Para reforzar

los conocimientos adquiridos acerca de git, la participante deberá subir cada archivo de manera individual, uno por cada *commit* y *push*.

5. Una vez ejecutado el push, la participante visualizará en su cuenta de *GitLab* que el repositorio se subió correctamente.
6. Cuando los pasos anteriores se hayan realizado sin complicaciones, se solicitará replicarlos para subir el proyecto a la plataforma *GitHub.*

Notas para apoyar la actividad:

- Se recomienda solicitarle a la participante buscar más comandos básicos de Git y que los ponga en práctica.

Links de apoyo (parte 1):

- JointDeveloper. (2017). *Sistemas de control de versiones, qué son y por qué amarlos.* https://medium.com/@jointdeveloper/sistemas-de-control-de-versiones-qu%C3%A9-son-y-por-qu%C3%A9-amarlos-24b6957e716e (Noviembre, 2020).
- Atlassian Bitbucket. (Sin fecha). *Software de control de versiones para equipos profesionales.* https://bitbucket.org/product/es/version-control-software (Noviembre, 2020).
- Platzi. (2019). *¿Qué es Git y GitHub?-Repositorios, ramas y mucho más* [video]. https://www.youtube.com/watch?v=DinilgacaWs (Noviembre, 2020).
- EDteam. (2019). *¿Qué es git y cómo funciona?* [video]. https://www.youtube.com/watch?v=jGehuhFhtnE (Noviembre, 2020).
- Rubio, J. (2019). *Qué es GIT y para qué sirve.* https://openwebinars.net/blog/que-es-git-y-para-que-sirve/ (Noviembre, 2020).
- Kjchints. (2020). *Git Explicado Fácilmente*. [video]. https://www.youtube.com/watch?v=2mxh3tgx71c (Noviembre, 2020).
- B, G. (2019). *¿Qué es GitHub y para qué se utiliza?* https://www.hoErnesto García. Tips de resolución de conflictos Git. https://medium.com/get-on-board-dev/tips-de-resoluci%C3%B3n-de-conflictos-git-b6a85b204f9bstinger.mx/tutoriales/que-es-github/ (Noviembre, 2020).
- Escuela Web. (2014). *Introducción a Git y Control de Versiones.* [video].

https://www.youtube.com/watch?v=SMWItqi3Y-0 (Noviembre, 2020).

- Xataka. (Sin fecha). *Qué es GitHub y qué es lo que le ofrece a los desarrolladores.* https://www.xataka.com/basics/que-github-que-le-ofrece-a-desarrolladores (Noviembre,2020).

Crear cuenta GitLab (parte 2):

- GitLab. (Sin fecha). *Regístrate*. https://gitlab.com/users/sign_up (Noviembre 2020).

Sistemas Git gratis:

- GitHub. (2016). *What is GitHub?* [video]. https://www.youtube.com/watch?v=w3jLJU7DT5E (Noviembre, 2020).
- LevelUpTuts. (2018). *What is GitLab?* [video]. https://www.youtube.com/watch?v=gbJUasioKiI (Noviembre,2020).
- Atlassian. (2016). *Bitbucket - the Git solution for professional teams*. [video]. https://www.youtube.com/watch?v=BD8xfCILcBs (Noviembre, 2020).
- Cleveroad. (2017). *Top 3 Source Code Repository Hosts* [video]. https://www.youtube.com/watch?v=AuLmDhY3Kjc (Noviembre,2020).

Configuración de un sistema git en línea:

- Fazt. (2018). *Git y GitHub | Curso práctico de Git y GitHub desde cero.* [video]. https://www.youtube.com/watch?v=HiXLkL42tMU (Noviembre, 2020).
- Laboratoria. (2017). *4. Cómo configurar Git* [video].
- https://www.youtube.com/watch?v=1RJHiHzUnk8 (Noviembre,2020).
- Git--everything-is-local. (Sin fecha). *1.6. Inicio- Sobre el control de versiones - Configurando git por primera vez.* https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Configurando-Git-por-primera-vez (Noviembre, 2020).

¿Cómo crear un repositorio en GitHub?

- Laboratoria. (2017). *2.Cómo crear un repositorio en GitHub* [video]. https://www.youtube.com/watch?v=Qy_rNobHRY0 (Noviembre,2020).
- Conociendo GitHub. (Sin fecha). *Aprendiendo a usar GitHub.*

https://conociendogithub.readthedocs.io/en/latest/data/dinamica-de-uso/#colaborar-en-un-proyecto-ajeno (Noviembre, 2020).

Comandos para subir un repositorio:

- Programación Web Full Stack. (2017). *Cómo subir el código de tu proyecto a GitHub.* https://www.ecodeup.com/como-subir-el-codigo-de-tu-proyecto-a-github/ (Noviembre,2020).
- Largo, E. (2018). *Cómo crear un repositorio local con Git y subirlo a GitHub*. [video]. https://www.youtube.com/watch?v=l2MdKm5RN3o (Noviembre, 2020).
- MitoCode. (2018). *Curso de Git y GitHub - 3 Primer repositorio y commit* [video]. https://www.youtube.com/watch?v=sZySm2Xh0vU (Noviembre,2020).
- Saldaña, N. (2018). *Cómo Crear Un Repositorio En GitHub Desde Terminal.* [video]. https://www.youtube.com/watch?v=tkQp1BvnzsY (Noviembre, 2020).
- Dudler, R. (Sin fecha). *Guit - La guía sencilla.* https://rogerdudler.github.io/git-guide/index.es.html (Noviembre,2020).

Trabajo colaborativo:

- Git. (Sin fecha). *5.1 Git en entornos distribuidos -Flujos de trabajo distribuidos.* https://git-scm.com/book/es/v2/Git-en-entornos-distribuidos-Flujos-de-trabajo-distribuidos (Noviembre, 2020).
- Pérez, A. (2010). *Git y cómo trabajar con un repositorio de código distribuido.* https://www.adictosaltrabajo.com/2010/07/12/git/ (Noviembre,2020).
- Atlassian Bitbucket. (Sin fecha). *Comparar workflows.* https://www.atlassian.com/es/git/tutorials/comparing-workflows (Noviembre, 2020).

Conflictos en git:

- García, E. (2019). *Tips de resolución de conflictos Git.* https://medium.com/get-on-board-dev/tips-de-resoluci%C3%B3n-de-conflictos-git-b6a85b204f9b (Noviembre,2020).

Links para aprender más:

- códigofacilito (Sin fecha). *Curso de Git Gratis*. https://codigofacilito.com/cursos/git (Noviembre, 2020).
- Learn Git Branching. (Sin fecha). *Aprendá a Branchear en Git.*

https://learngitbranching.js.org/?locale=es_AR (Noviembre, 2020).

Actividad 1: Programación en Python

<table>
<tr><td colspan="2">Aprendizaje esperado: Ejecutar un programa que muestre un mensaje en la consola de texto.</td><td>Duración de la actividad: 3 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación.</td></tr>
<tr><td>● Hojas de papel, cartulina, papel bond o papel craft.
● Colores o plumones.
● Python 3.x.x
● Anexo MPT1A1_Anexos.pdf</td><td>Evidencia:
1. Cartel o infografía.
2. Mensaje importado en pantalla.</td><td>Retroalimentación:
● El cartel deberá contener información acerca de la utilidad de la programación; la definición y la utilidad de Python; y la aportación de dos mujeres en el campo de la programación. Además, deberá incluir algún mensaje que invite a las mujeres a aprender programación con Python y a unirse a la Escuela de Código. No deberá utilizar lenguaje ofensivo, que incite a la violencia, o que reproduzca estereotipos de género.
● El mensaje deberá ser importado a través del modo interactivo de Python, con la leyenda Hola mujeres.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Avioncito programador.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.</td></tr>
</table>

La participante jugará el juego del avioncito con algunas modificaciones, las cuales le motivará a reflexionar acerca de qué es la programación y qué son los lenguajes de programación. Para lo anterior, seguirá las siguientes instrucciones (se recomienda utilizar el anexo *MPT1A1_Anexos.pdf)*:

1. Se jugará con la participante en caso de que no esté disponible alguna otra persona, ya que el juego requiere 2 jugadoras.
2. Se dibujarán en el suelo o en alguna cartulina las casillas del 1 al 10, en caso de desconocer el orden de las casillas, se recomienda remitirse a la plantilla 4 del anexo ya mencionado.
3. Se establecerán los roles de las jugadoras: rol de programadora y rol de computadora. La primera dictará una secuencia de instrucciones a seguir que la segunda deberá interpretar y ejecutar.
4. La jugadora con rol de programadora deberá compartir con la jugadora con rol de computadora, la secuencia que se sugiere en el anexo de la presente actividad, pero sin utilizar las palabras, únicamente lenguaje corporal.
5. La jugadora con rol de computadora deberá interpretar los movimientos de su compañera y ejecutar lo que comprenda que significan. Tendrá un límite de doce segundos para realizar la tarea.
6. En una segunda ronda, la jugadora con rol de programadora compartirá la misma secuencia de instrucciones, pero ahora a través de lenguaje gesticular (gestos con el rostro). La jugadora con rol de computadora deberá nuevamente interpretar y ejecutar las instrucciones, pero ahora contará con once segundos para hacerlo.
7. En una tercera ronda, se compartirá la misma secuencia de forma normal, en español, tal y como se enuncia en la plantilla 3 del *MPT1A1_Anexos.pdf*. La jugadora con rol de computadora tendrá un tiempo límite de diez segundos para interpretar y ejecutar todas las instrucciones.
8. Al terminar, intercambiarán los roles y realizarán nuevamente las 3 rondas con los pasos descritos.
9. Una vez finalizado el juego, se realizará un diálogo de retroalimentación recuperando las experiencias del juego. Se sugiere utilizar las siguientes preguntas como guía de la conversación:
    1. ¿Cómo te sentiste con el juego? ¿Se te dificultó algo?
    2. ¿Cuál era la función del rol de programadora y cuál la función como rol de computadora?
    3. ¿En qué consiste entonces la programación?
    4. ¿En cuál de los 3 lenguajes fue más fácil interpretar las instrucciones?
    5. ¿Qué es entonces un lenguaje de programación?
    6. ¿Cuál es la diferencia entre un algoritmo y un lenguaje de programación? (Aclarar a la participante la diferencia).
    7. ¿Cuál es la diferencia entre compilar e interpretar? (Aclarar a la participante la diferencia).
    8. ¿Crees que existe un solo lenguaje de programación?

10. Para concluir la primera parte, se le solicitará a la participante que realice un cuadro comparativo (como el que se muestra a continuación) sobre algunos lenguajes de programación (entre 4 y 6 lenguajes) que encuentre por medio de una búsqueda en internet.

Lenguaje de programación	Fecha de lanzamiento	Persona o entidad que lo desarrolló	Características más sobresalientes	En qué lenguaje anterior está inspirado

Segunda parte: Aprendiendo y compartiendo.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante tendrá un primer acercamiento a *Python*. Para ello, se le preguntará si dentro de los lenguajes que utilizó para elaborar el cuadro de la primera parte de la actividad, se encuentra *Python*. De ser así, se le solicitará una breve exposición de lo que encontró acerca de este lenguaje. De no haber colocado *Python* en dicho cuadro, se le solicitará añadirlo, para ello, deberá ver al menos tres videos en donde se explique brevemente qué es y cuáles son sus características principales.
2. Además de las categorías del cuadro, la investigación de la participante deberá cubrir los temas:
    a) *Python* para celulares (*IOS* y *Android*) y *Python* para computadoras (*Windows, Mac* o *Linux*).
    b) Modo interactivo de *Python.*

3. Posteriormente, procederá a aclarar si la información que encontró es correcta o no, y completar los datos que se crean necesarios.
4. Realizado el acercamiento conceptual a *Python*, se le solicitará a la participante la creación de un cartel o infografía, de forma digital o a mano, que se pegará en el PILARES. En él deberá exponer:
    1. ¿Para qué sirve la programación?
    2. ¿Para qué sirve *Python*?
    3. Incluir el nombre y aportación de dos mujeres en programación.

    La intención será invitar a más mujeres a que aprendan programación con *Python* y que se unan a la Escuela de código.

Tercera parte: Mensaje en modo interactivo.
Sugerencia de tiempo invertido: 30 minutos.

1. La participante imprimirá su primer mensaje en *Python.* Para ello, se le solicitará retomar la información encontrada acerca del uso del modo interactivo en *Python* en la parte dos de la actividad. A continuación deberá imprimir un “Hola mujeres”.
2. Al finalizar, se llevará a cabo un diálogo para recuperar cuáles fueron sus sentimientos al ver impreso en la pantalla, su primer programa en *Python*, y qué espera de las siguientes actividades del taller.

Links para aprender más:

- EDteam. (2020). *Tipos de lenguaje de programación* [video]. https://www.youtube.com/watch?v=gFMMmi-EYEM (Mayo, 2020).
- Unidad de Apoyo para el Aprendizaje, CUAED, UNAM. (Sin fecha). *Lenguajes de Programación*. https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html (Mayo, 2020).
- Conde, J. (2014). *Qué es Python* [serie de videos]. https://www.youtube.com/watch?v=oKQMoxJR5uk&list=RDQMQ5aJcvxnqDA&start_radio=1 (mayo, 2020).
- TokioSchool. (Sin fecha). *Conoce la sintaxis en Python*. https://www.tokioschool.com/noticias/conoce-sintaxis-python/ (Mayo, 2020).

- Yeeply. (2019). *Lenguajes de programación más usados según el tipo de desarrollo* https://www.yeeply.com/blog/lenguajes-de-programacion-mas-usados/ (Mayo, 2020).

Actividad 2: Conviviendo con una serpiente

<table>
<tr><td colspan="2">Aprendizaje esperado: Ejecutar un programa que muestre un mensaje en la consola de texto.</td><td>Duración de la actividad: 5 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Python 3.x.x
● Pycharm.
● Plantilla MPT1A2_LenguajePython.pdf
● Cualquier medio físico o digital para resolver la plantilla.</td><td>Evidencia:
1. Preguntas de reflexión sobre la alianza entre mujeres.
2. Cuadro comparativo consola de computadora vs. Pycharm.
3. Plantilla MPT1A2_LenguajePython.pdf respondida.
4. Código que integra al proyecto “MPT1A2_Programas”.

Producto:
1. Script “miMensaje.py” utilizando la función print.
2. Script “miMensajeConMain.py” utilizando la función main.
3. Script “horaFecha.py” importando el módulo datetime.</td><td>Retroalimentación:
● En las preguntas discusión acerca de alianzas entre mujeres, evitar los monosílabos o respuestas muy cortas; sugerir los principios del apoyo entre mujeres mencionados en la primera parte de esta actividad.
● En el cuadro comparativo, para consola de computadora: Revisar que el proceso o pasos que se pongan en el cuadro comparativo correspondan a la ejecución del Script.
● Revisar que las respuestas en los recuadros de la plantilla, las escriban con sus propias palabras; la definición deberá estar completa y corresponder a cada parte del código que se está describiendo.
● Verificar que se presenta de forma correcta los mensajes asociados al proyecto “MPT1A2_Programas”.

Evaluación:</td></tr>
</table>

		● Cuidar que la sintaxis en cada *script* no genere ningún error en su ejecución y ejecute lo que se está pidiendo en cada *script*.

Desarrollo de la actividad:

Primera parte: Nuevos tiempos, nuevas alianzas.

Sugerencia de tiempo invertido: 1 hora.

Es importante fomentar y hacer énfasis en que entre las mujeres se generan alianzas y se llega a un fin común, por lo tanto, para dar inicio:

1. Se le hará saber a la participante que va a construir un juego de ajedrez a través de *Python*.
2. La participante desarrollará a lo largo de las actividades la habilidad de programar, así como reconocer los elementos del ajedrez desde el apoyo entre compañeras.

Entendiéndose este como:

- Reconocer a la otra u otras como iguales.
- Mantener el respeto y reconocimiento por la otra u otras.
- Encontrar puntos en común para trabajar sobre ellos.
- Trabajo equitativo para el logro de objetivos.

Este proceso puede hacerse haciendo equipo entre compañeras para apoyarse de manera que entre todas reconozcan los elementos del ajedrez, o bien, para tratar otros temas relacionados con la programación; si se prestan las condiciones, se sugiere que se haga en duplas, de participante a participante, en caso de no ser posible, la tallerista podrá conjuntar un equipo con la participante.

1. En un espacio de 10 minutos aproximadamente, reflexionará las siguientes preguntas:
    - ¿Consideras que hay menos mujeres que representen el juego del ajedrez?¿Por qué?
    - ¿Consideras que sería más fácil aprender a programar y a jugar ajedrez con el apoyo de otras compañeras?¿Por qué?
    - ¿Crees que el apoyo entre mujeres nos ayuda a resolver problemas conjuntos? ¿Por qué?
2. Anotará las conclusiones a las que llegó y las comentará ya sea de participante a participante, o de participante a tallerista, según sea el caso; es importante llevar a la participante a la reflexión de las preguntas usando su experiencia.

Segunda parte: No todas las serpientes muerden.

Sugerencia de tiempo invertido: 2 horas.

La participante resolverá la plantilla *LenguajePython* (disponible en el anexo *MPT1A2_Anexos.pdf)*, que servirá como introducción a la sintaxis del lenguaje de programación *Python*, para posteriormente programar sus propios *scripts*.

1. La plantilla consiste en un bloque de código con sus partes nombradas, por ejemplo: comentario, importación de módulo, función principal, *print,* etcétera.
2. Como primer acercamiento, reflexionará acerca de las posibles funciones que podrían tener cada una de las partes señaladas, de igual manera se comentarán con la tallerista.
3. Una vez observada la plantilla, realizará una búsqueda en internet sobre cada una de las partes que señala la plantilla y redactará la función de cada una.
4. Una vez finalizado el ejercicio anterior, la participante abrirá y explorará el entorno de desarrollo integrado (IDE) *pycharm*, podrá discutir en parejas o con la tallerista qué es lo que observa y qué funciones tendría cada parte.
5. Como cierre, apoyándose de los conocimientos obtenidos en el *Módulo 0*, la participante realizará una tabla comparativa donde se explique el proceso para ejecutar los archivos *Python* en la consola de la computadora, y en el editor de textos.

Ejemplo:

Ejecución en la consola de la computadora	Ejecución en la consola de pycharm
Para ejecutar en la consola se debe….	Para ejecutar en pycharm se debe...

Tercera parte: ¡A programar!

Sugerencia de tiempo invertido: 2 horas.

La participante programará tres *scripts* en *Python* apoyándose del IDE *pycharm.* Los archivos *Python* deberán guardarse en un proyecto con nombre “MPT1A2_Programas”.

1. En el primer *script* imprimirá un mensaje sencillo por medio de la función *print*, el mensaje a imprimir sugiere ser “Nosotras también jugamos ajedrez”, el archivo se llamará “miMensaje.py”.
2. En el segundo *script*, imprimirá un mensaje más largo utilizando la función *main***,** el mensaje deberá imprimir el número de piezas de ajedrez de ambos equipos, el número de casillas en el tablero de ajedrez y por último, listará el nombre de las piezas del juego, nombrará el archivo “miMensajeConMain.py”.
3. En el tercer *script* importará el módulo *datetime* para imprimir la hora y fecha actual de la computadora, deberá hacer uso de la función *main* para la impresión del mensaje, el archivo se llamará “horaFecha.py”.

4. La ejecución se visualizará en la consola de *pycharm*.
5. Al finalizar este segmento, deberá subir el proyecto creado a la carpeta de evidencias en la nube.

Nota:

- Se le recomendará a la participante descargar e instalar *Python* y *Pycharm* en su computadora personal.

Links de apoyo:

- JetBrains. (2020). *Descargar Pycharm.* https://www.jetbrains.com/es-es/pycharm/download/#section=linux (Mayo, 2020).
- Python. (2020). Descargar *Python.* https://www.python.org/downloads/ (Mayo, 2020).
- Stiglitz, R. (2016). *[Tutorial PyCharm] 2. Crear y ejecutar en PyCharm.* https://travelerisallama.blogspot.com/2016/06/python-2-crear-y-ejecutar-en-pycharm_29.html (Mayo, 2020).
- Tokioschool. (Sin fecha) *Conoce la sintaxis en Python.* https://www.tokioschool.com/noticias/conoce-sintaxis-python/ (Mayo, 2020).
- CÓDIGO FUENTE. (2018). *Sintaxis en Python.* https://www.codigofuente.org/sintaxis-en-python/ (Mayo, 2020)
- MC Libre. (2020). *Salida por pantalla: la función print()* https://www.mclibre.org/consultar/python/lecciones/python-salida-pantalla.html (Mayo, 2020).

Actividad 3: Hablando con Python

<table>
<tr><td colspan="2">Aprendizaje esperado: Ejecutar un programa que muestre un menú de opciones para iniciar, reiniciar, pausar o salir de un juego de ajedrez.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel ó cartulina.
● Bolígrafos.
● Plumones.
● Tijeras.</td><td>Evidencia:
1. Script “trivia.py”.
2. Script “preguntasRespuestas.py”
3. Script “ajedrezMovimientos.py”</td><td>Retroalimentación:
● En los scripts “trivia.py”, “preguntasRespuestas.py”, “ajedrezMovimientos.py” y cuidar que la estructura de los programas contengan la función main donde deberá ser escrito el código.
● En el script “trivia.py” no promover estereotipos de género en las respuestas impresas.
● En el script “preguntasRespuestas.py” revisar que las palabras ordenadas respeten el orden sintáctico y que el código no marque error en el interpretador.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Si ya sé imprimir, ¿cómo leo los mensajes?
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. En esta parte lo ideal es trabajar de participante a participante, sin embargo, de no ser posible, la participante podrá realizar el ejercicio con la tallerista, o bien, en solitario. Reconocerá la estructura sintáctica de la función input(), es importante dejar</td></tr>
</table>

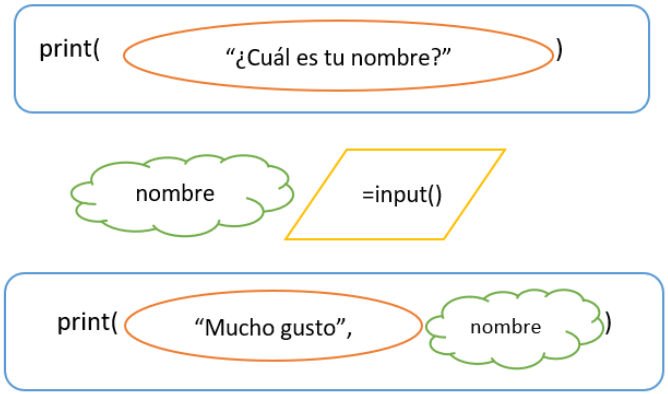
que haga el ejercicio por sí misma, ya sea realizando una búsqueda de dicha función o por razonamiento propio. Deberá elaborar 15 tarjetas:

- En forma de paralelogramo, una tarjeta con la leyenda *=input()*.
- En forma de rectángulo con las esquinas redondeadas, dos tarjetas con la leyenda *print()*, deberá dejar un espacio considerable de modo que quepan el texto de la impresión.
- En forma de óvalo horizontal, tres tarjetas formulando preguntas sobre algún tema, por ejemplo, *"¿Cuál es tu nombre?"*,*"¿Cuál es tu color favorito?"*, *"¿A dónde te gustaría viajar?"*; de la misma forma otras tres tarjetas que respondan a las preguntas formuladas, recordar que todas deben incluir comillas.
- En forma de nube, seis tarjetas que nombren cada variable que almacena la respuesta en el *input*; serán seis, ya que se esperan tres variables y cada variable deberá repetirse dos veces, una para guardar el *input()*, y la otra para la impresión.

  Se sugiere nombrar las variables de acuerdo a lo que se pregunta, por ejemplo, si la pregunta es *"¿Cuál es tu color favorito?"*, el nombre de la variable sería *"color"*.

2. La participante construirá tres códigos ordenando todas las tarjetas, usando la función *print()* para imprimir un mensaje y una función *input()* para la entrada por teclado, misma que se guardará en una variable, para después imprimirse con otro *print()*. Deberán alternar las preguntas y respuestas para armar todos los códigos.

   Ejemplo:



3. La participante programará un *script* nombrado “preguntaRespuesta.py” correspondiente a cada código armado con las tarjetas, los archivos *Python* deberán guardarse en un proyecto nombrado “MPT1A3_Preguntas”. Cada *script* deberá implementar una función *main* y se deberá comprobar que la sintaxis de todo el código no presente errores.

Segunda parte: Toda acción tiene una reacción.
Sugerencia de tiempo invertido: 1 hora.

1. Se usará una analogía como ejemplo para entender el principio de los condicionales *if-elif-else*; cada pieza en un tablero de ajedrez tiene un movimiento específico. La participante escribirá, primero en papel, el pseudocódigo de dichos movimientos con la estructura “Si”, “De lo contrario”, “Entonces”, como elementos ilustrativos a la analogía. Se tendrá que hacer una búsqueda de cómo se mueven las piezas en caso de no conocer los movimientos.

   Ejemplo:

   pieza = "caballo"
   Si pieza *es igual* a “alfil” **entonces**
   “se mueve en diagonal”
   De lo contrario, si pieza *es igual* a “torre” **entonces**
   “se mueve en cruz”
   De lo contrario, si pieza *es igual* a “caballo” **entonces**
   “se mueve en L”
   ...

   De lo contrario, entonces
   "el nombre de la pieza no corresponde a ninguna pieza del ajedrez"

2. De acuerdo a la analogía construida, la participante realizará una búsqueda en internet acerca de las condicionales *if-elif-else*. Con base en la información recolectada, reemplazará en un *script* de *pycharm* los elementos “Si”, “De lo contrario, si”, “De lo contrario”, “Entonces” y “es igual” de su analogía, por los condicionales *if*, *elif* y *else*, y por los símbolos *:* y ==. Nombrará el archivo “ajedrezMovimientos.py”, y lo guardará en el proyecto *MPT1A3_Preguntas*.
   - “*if*” es equivalente a “si”.
   - “*elif*” es equivalente a “de lo contrario, si”.

- “*else*” es equivalente a “de lo contrario“.
- “:” es equivalente a “entonces”.
- “==” es equivalente a “es igual”.

Tercera parte: Hoy ando muy preguntona.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante programará un *script* nombrado “trivia.py”, utilizando lo revisado hasta el momento: condicionales *if*, *elif*, *else*; *input()* y *print()*, el archivo deberá guardarse en el proyecto *MPT1A3Preguntas*.
Los enunciados serán las cuestiones a resolver y dependiendo de las respuestas que se introduzcan, se deberán imprimir las leyendas “Respuesta correcta” o “Respuesta incorrecta” según sea el caso, e incluirá el por qué son correctas o incorrectas.

Ejemplo:

- El rosa es para niñas y el azul para los niños.
    - Verdadero
        - “Respuesta incorrecta, la idea rosa es para niña y azul es para niño responde a un estereotipo que asocia características dependiendo del sexo con el que nació un infante, sin embargo, todos los colores los puede usar cualquier persona”
    - Falso
        - “Respuesta correcta, cada persona puede usar el color que guste independientemente del sexo o género con el que se identifique”
- Las mujeres no pueden trabajar en el campo de la tecnología, los hombres sí.
    - Verdadero
        - “Respuesta incorrecta, tanto hombres como mujeres tienen la capacidad de desempeñar actividades en la tecnología, sin embargo, las últimas no han tenido la oportunidad de destacar en un campo mayormente poblado por hombres”
    - Falso
        - “Respuesta correcta, las mujeres tienen toda la capacidad para trabajar en el campo de la tecnología”
- Los hombres programan mejor que las mujeres.
    - Verdadero

    - “Respuesta incorrecta, de hecho la primera programadora fue una mujer: Ada Lovelace”
- Falso
    - “Respuesta correcta, el sexo no determina si alguien es mejor o peor haciendo alguna actividad”

Notas para apoyar la actividad:

- Aclarar que la estructura de código con las tarjetas va dentro de una función *main( )*.
- De momento, la participante no sabrá con exactitud qué está trabajando variables, para evitar confusiones.
- Es importante fomentar que las participantes realicen las actividades por sí mismas sin mucha ayuda, sin embargo, guiarlas si es que se llegasen a confundir.

Links para aprender más:

- MC Libre. (Sin fecha). *Entrada por teclado: la función input().* https://www.mclibre.org/consultar/python/lecciones/python-entrada-teclado.html (Mayo, 2020).
- COVANTEC. (Sin fecha). *Entrada/Salida en Python.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion7/entrada_salida.html (Mayo, 2020).
- Pythones.net. (Sin fecha). *IF, ELSE, ELIF: Aprende los condicionales en Python con ejemplos!* https://pythones.net/if-else-elif-condicionales/ (Mayo, 2020).
- MC Libre. (Sin fecha). *if ... elif ... else ...* https://www.mclibre.org/consultar/python/lecciones/python-if-else.html (Mayo, 2020).
- COVANTEC. (Sin fecha). *Condicional if.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion4/condicional_if.html (Mayo, 2020).
- Parzibyte. (2018). *Leer e imprimir datos en Python con input y print.* https://parzibyte.me/blog/2018/10/17/leer-imprimir-datos-python-input-print/ (Mayo, 2020).

Actividad 4: Mi primera ventana con PyGames

<table>
<tr><td colspan="2">Aprendizaje esperado: Ejecutar un programa que muestre un mensaje en la consola de texto.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr>
<td>

Python 3.x.x
PyGame 2.0.0 dev 7
Anexo MPT1A4_Anexos.pdf
Anexo MPT1A4_ImagenPieza_BlackKnight.png
Documentación de PyGame.
Hoja milimetrada de 5mm y 10 mm.

</td>
<td>
Evidencia:

Anexos MPT1A4_Anexos.pdf resueltos.
Archivo ventana.py.

Productos:

Archivo tablero.py

</td>
<td>
Retroalimentación:

Para evaluar la resolución de los anexos MPT1A4_Anexos.pdf, las plantillas 2, 3, 4 y 6 deberán estar respondidas. En lo que respecta a la plantilla 6, deberá asegurarse que los ejemplos propuestos por la participante cuenten con la sintaxis del ciclo while como se ejemplifica en la plantilla 5.
El archivo ventana.py deberá haber sido desarrollado con el siguiente orden de sentencias:

Para agregar color a un objeto:
pygame.fill()
Actualiza todos los cambios en la interfaz:
pygame.display.flip()
Inicializa la interfaz gráfica:
pygame.init()
Para cambiar el encabezado de la interfaz:
pygame.display.set_caption()
Para cambiar el icono de la ventana:

pygame.display.set_icon()
pygame.display.set_mode()
pygame.event.get()

Quitar la interfaz gráfica:
pygame.quit()
</td>
</tr>
</table>

		Actualiza los cambios en una parte de la interfaz gráfica: ● pygame.update() Para carga y manejo de imágenes: ● pygame.image.load() Para cambiar el tamaño de la interfaz: ● pygame.transform.scale() Dibujar imágenes sobre la interfaz: ● blit() **Evaluación:** ● En lo que respecta al archivo *tablero.py*, deberá desarrollarse con las siguientes sentencias: ● pygame.display.set_caption() ● Ciclo *while()* ● *pygame.draw.rect()* ● pygame.image.load() ● pygame.transform.scale() ● Además, deberá contar con las siguientes funciones: *dibujaTablero(pantalla), main()* y *llenaTablero(pantalla)*. Dichas funciones deben ser creadas por la participante.

Desarrollo de la actividad:

Primera parte: Color al cuadrado.
Sugerencia de tiempo invertido: 30 minutos.

La participante asimilará los conceptos de píxeles, resolución de imagen y sistemas de código de color a través de colorear las figuras que se presentan en el *MPT1A4_Anexos.pdf*.

1. Se solicitará resolver la plantilla 3 del anexo *MPT1A4_Anexos.pdf*. En dicha plantilla se encuentran dibujadas las seis figuras básicas del juego de ajedrez en una hoja milimetrada de 10 mm. Cada figura está representada por un número, al cual le corresponde un color específico enunciado con un código, que deberá buscar en internet para identificar qué color es. La participante deberá colorear cada figura y escribir el nombre de esta en cada línea de la plantilla. De dificultarse la utilización de las plantillas, se recomienda comprar una hoja milimetrada de 10 mm y copiar cada figura en ella para realizar el ejercicio.
2. Posteriormente, se solicitará a la participante realizar el mismo procedimiento para colorear la plantilla 4 del *MPT1A4_Anexos.pdf*. En dicha plantilla, se encuentran dibujadas las seis figuras básicas del juego de ajedrez en una hoja milimetrada de 5 mm.
3. Una vez que la participante termine de colorear las dos plantillas, se le solicitará responder las siguientes preguntas por medio de un diálogo de retroalimentación:
    - ¿En qué plantilla se ven más claras las figuras? ¿A qué se debe?
    - ¿Qué son los píxeles?
    - ¿Qué es la resolución de imagen?
    - ¿Qué significan las claves de los colores?
    - ¿Cuántos sistemas de código de color existen?

Segunda parte: ¿Mientras qué hago?
Sugerencia de tiempo invertido: 30 minutos.

La participante conocerá el funcionamiento básico de un ciclo *while* en programación a través de la resolución de las plantillas 5 y 6 del anexo *MPT1A4_Anexos.pdf*. Para ello, se le solicitará la lectura del siguiente ejemplo (también disponible en la plantilla 5):

```
while(mujeres formen alianzas)
    print(“Las mujeres son más fuertes”)
    print(“Las mujeres pueden ayudarse”)
    print(“Las mujeres pueden apoyarse”)
print(“Las mujeres deben formar alianzas, eso las fortalece”)
```

Una vez revisado el ejemplo, se solicitará a la participante que en la siguiente plantilla presente 2 ejemplos escritos en pseudocódigo, y posteriormente traducidos a código según la sintaxis del ciclo *while*. En caso de ser necesario, se motivará a la participante a realizar una búsqueda en internet para aclarar cualquier duda durante la creación de sus ejemplos.

Tercera parte: Mi primera ventana con PyGames.
Sugerencia de tiempo invertido: 1 hora.

La participante desarrollará su primera ventana de interfaz gráfica para crear un videojuego con *Python.* Para ello se le solicitará realizar las siguientes instrucciones, apoyándose en la documentación de *PyGames*:

1. Creará un archivo llamado “ventana.py”.
2. Importará la biblioteca para utilizar *PyGame.*
3. Creará un programa que permita visualizar una ventana al ejecutar el programa.

A continuación deberá realizar algunas modificaciones para observar cambios en dicha ventana. Para ello:

1. Utilizará un ciclo *while* para ejecutar la ventana hasta que decida cerrarse con el botón de cerrar ubicado en la parte superior derecha de una interfaz.
2. Cambiará el encabezado de la interfaz.
3. Cambiará el icono del encabezado de la interfaz.
4. Cambiará el color de la interfaz.
5. Agregará una imagen de fondo que abarque todo el tamaño de la interfaz.
6. Agregará una imagen que se desplace horizontalmente sobre la interfaz.
7. Hará que la misma imagen del punto 6 se desplace verticalmente sobre la interfaz.

Para cada uno de los cambios realizados, la participante deberá ejecutar el programa con el fin de visualizarlos en cada instrucción y comprender el objetivo de las funciones que se utilizan.
En caso de no presentar ningún error de sintaxis o lógica al momento de ejecutar el programa en consola y poder visualizar todos los cambios solicitados en los puntos anteriores, la participante añadirá el código de la actividad a la carpeta de evidencias.

Cuarta parte: Diseñando mi tablero de ajedrez.

Sugerencia de tiempo invertido: 2 horas.

Se le solicitará a la participante seguir las siguientes instrucciones:

1. Creará un archivo llamado *"*tablero.py*"*.
2. Importará la biblioteca para utilizar PyGame.
3. Cambiará el encabezado de la interfaz por "Mi ajedrez en *Python*".
4. Se le proporcionará a la participante la sintaxis de las funciones en *Python* (este tema se verá a profundidad en la actividad 10), y con base en esta, se le pedirá crear una función *main()* que no reciba parámetros y en la cual se encuentre el código correspondiente a la ejecución de la interfaz dentro de un ciclo *while*, parecida a la que se desarrolló en la parte anterior de la presente actividad. Dentro de esta función deberán invocarse a la función *dibujaTablero(pantalla)*.
5. Creará una función "dibujaTablero(pantalla)" que reciba como parámetros el nombre de la interfaz en donde se mostrará el tablero. La función deberá encargarse de dibujar el tablero de ajedrez con ayuda de la función *pygame.draw.rect.* Será importante tomar en cuenta que no deberá hacerse uso de ningún ciclo de control (*if*-*else*/*while*/*for*) para la implementación de esta función. El tablero deberá constar de una cuadrícula de 8x8 de color negro y blanco, alternadamente.
6. Creará una función *llenaTablero(pantalla)* que reciba como parámetros el nombre de la interfaz en donde se dibujó el tablero. La función deberá colocar en el tablero una de las piezas para el juego, a través de la inserción de una imagen sobre la casilla que le corresponda. Si la imagen no contase con el tamaño adecuado, deberá escalarse con una función de PyGame. Se sugiere utilizar la imágen que se brinda en el anexo *MPT1A4_ImagenPieza*_BlackKnight.png

En caso de no presentar ningún error en el programa, se realizará un diálogo de retroalimentación en donde se recuperará qué se le dificultó y si aún tienen alguna duda. Posteriormente, con la intención de motivar a la participante, se le recomendará mostrar a alguna mujer cercana a ella la ventana gráfica de su tablero y su pieza o piezas de ajedrez, a la vez que explica su experiencia o el procedimiento que tuvo que llevar a cabo para realizar su programa.

Notas para apoyar la actividad:

- La participante subirá el script *tablero.py* a su repositorio *juegoDeAlianzas_Ajedrez* de GitHub.
- Para la cuarta parte de la actividad, no será necesaria la total comprensión del uso y sintaxis de las funciones en *Python* (este tema se verá más adelante en la actividad 10).

- Para la cuarta parte de la presente actividad, en caso de utilizar una imagen diferente a la del anexo *MPT1A4_ImagenPieza_BlackKnight.png*, asegurarse de que ésta cuente con licencia creative commons o que no cuente con derechos de autor, de no ser así, no podrá ser utilizada.
- Se asegurará de que en el punto 5 de la cuarta parte de la presente actividad, la participante utilice los colores para el tablero en formato RGB.
- Se recomienda motivar que la participante comprenda la diferencia entre la elaboración manual de cada casilla y haciendo uso del ciclo *while*.

Links para aprender más:

- freeCodeCamp. (2019). *Pygame Tutorial for Beginners - Python Game Development Course.* https://www.youtube.com/watch?v=FfWpgLFMI7w (Julio, 2020).
- Pygame. (Sin fecha). *Documentation*. https://www.pygame.org/docs/ (Junio, 2020).

Actividad 5: Variables de una dimensión y sus operaciones

<table>
<tr><td colspan="2">Aprendizaje esperado: Almacenar en variables el nombre, el tipo, el bando y la posición de las piezas de ajedrez.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos.</td><td>Evidencia/producto.</td><td>Retroalimentación/Evaluación.</td></tr>
<tr><td>● Plantilla MPT1A5_Anexos.pdf</td><td>Evidencia:
1. Plantilla MPT1A5_Anexos.pdf resuelta.
2. Cuadro comparativo MPT1A5_AcercadelasVariables resuelto.

Producto:
1. Proyecto MPT1A5_Variables con los siguientes scripts
○ misVariables.py
○ erroresEnVariables.py
○ misPiezasDeAjedrez.py
2. Modificaciones del proyecto tablero.py</td><td>Retroalimentación:
● La plantilla deberá responderse completa y de acuerdo a lo que se solicita en las instrucciones de la misma.
● En el cuadro comparativo deberá estar completo, y revisar que la información sea sintética y completa.

Evaluación:
● Cuidar que la sintaxis en cada script solicitado no genere ningún error en su ejecución y deberán ejecutar lo solicitado en cada caso.
● En el caso del script erroresEnVariables.py, las correcciones deberán corresponder a las reglas para nombrar variables anteriormente revisadas; los comentarios de corrección de los nombres de las variables deberán explicar los motivos por los que se generó un error al momento de ejecutar el script, tomando en cuenta los parámetros vigentes en Python para nombrar variables.</td></tr>
</table>

Desarrollo de la actividad:

Primera parte: Las cajas organizadoras de una serpiente.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante realizará una búsqueda acerca de las variables, guiándose de las siguientes cuestiones:

- ¿Qué son las variables en *Python*?
- ¿Cuáles son sus características?
- ¿Cómo se declara una variable en *Python*?
- ¿Cómo se asigna un valor a una variable en *Python*?
- ¿Cuáles serían las reglas para nombrar a una variable?

El resultado de la búsqueda se materializará en el siguiente cuadro comparativo *MPT1A5_AcercadelasVariables*; se podrá emplear como material de consulta en el futuro.

¿Qué es una variable?		
¿Cómo nombrar a mis variables?		
Tipos de variables		
Enteras	*Flotantes*	*Cadenas*

2. La participante resolverá los anexos *MPT1A5_Anexos.pdf* para familiarizarse con la notación algebraica del tablero de ajedrez. Deberá primeramente completar las casillas vacías de la plantilla 2, con los números y las letras que correspondan.

Posteriormente, recortará las 3 figuras de mujeres haciendo alianzas, para pegarlas sobre la misma plantilla 2, cada una en la casilla que prefiera. Por último, en la plantilla 3, la participante deberá escribir haciendo uso de las coordenadas con letra y número, la casilla en la que localizó cada figura.

Segunda parte: Aprende a utilizar las cajas de una serpiente.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante creará un nuevo proyecto en pycharm con nombre "MPT1A5_Variables". Se recomienda realizar los siguientes *scripts*, sin embargo, se dejará a criterio incorporar uno o más ejercicios si así lo requieren las necesidades de las participantes. Se motivará a la participante a averiguar cómo resolver los puntos enunciados conjuntando búsquedas en internet y práctica.
Un *script* con nombre "misVariables.py":

- Creará una variable de nombre "numeroEntero" y asignará su número favorito, se asegurará de que sea un número entero.
- Creará una variable de nombre "numeroFlotante" y asignará su estatura, se asegurará que sea un número con punto decimal.
- Creará una variable de nombre "miNombre" y asignará el nombre propio.
- Imprimirá cada variable en pantalla y con la función *type()*; revisará el tipo de cada una de las variables.
- Actualizará el valor de la variable "numeroEntero" haciendo una nueva asignación sumando su segundo número favorito, también deberá restar, multiplicar y dividir, volver a imprimir la variable cuantas veces sea necesario para observar las actualizaciones.
- Hará lo mismo con la variable "numeroFlotante", esta vez utilizando números con punto decimal.
- Creará una nueva variable asignándole su apellido y utilizará el operador de concatenación *+* para concatenar esta variable y la variable "miNombre" e imprimirla en pantalla.
- Utilizará el operador de repetición ***** para repetir la cadena almacenada en la variable "miNombre" e imprimirla en pantalla.

Un *script* con nombre "erroresEnVariables.py"*:*

- Creará una variable con nombre de una palabra reservada que haya sido vista en actividades anteriores, por ejemplo *while* o *print* y asignarle un valor, observar qué sucede. Probará con diferentes palabras reservadas.
- Creará las siguientes variables y observará qué sucede:

```
mivariable = "Esta es mi variable"
mi-variable = "Esta es mi otra variable"
mi variable = "Esta es mi última variable"
```

- Después se observarán los errores posibles y por qué ocurren, se deberá corregir el nombre de las variables de acuerdo a los criterios para nombrar variables anteriormente revisados; a modo de comentario describir porqué ocurre un error en cada uno de los casos.

Tercera parte: Almaceno mis piezas de Ajedrez en cajas.
Sugerencia de tiempo invertido: 1 hora.

Se le solicitará a la participante crear un nuevo *script* con nombre "misPiezasDeAjedrez.py" guardándolo en su proyecto *MPT1A5_Variables,* en el cual realizará las siguientes instrucciones:

1. En la función principal del programa creará 32 variables de tipo entero en donde asignará la posición inicial del renglón donde se encuentra cada pieza de ajedrez, en notación algebraica. El nombre de las variables será alusivo al nombre de la pieza y el renglón, por ejemplo:

   reinaBlancaRenglon = 1

2. Creará 32 variables de tipo carácter en donde asignará la posición inicial de la columna donde se encuentra cada pieza de ajedrez, en notación algebraica.El nombre de las variables será alusivo al nombre de la pieza y la columna, por ejemplo:

   reinaBlancaColumna = 'e'

3. Creará 6 variables y le asignará el nombre de cada pieza.
4. Creará 2 variables y le asignará el bando al que pertenecen.

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar los *scripts*, se le motivará para que realice una búsqueda en internet acerca de los elementos necesarios para encontrar o descubrir las soluciones. Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la

carpeta de evidencias correspondiente.

Links para aprender más:

- Docs.python. (2020). *Documentación de Python-3.8.6.* https://docs.python.org/es/3.8/ (Julio, 2020).
- Mc Libre. (2019). *Variables.* https://www.mclibre.org/consultar/python/lecciones/python-variables.html (Julio, 2020).
- Lozano, J. (Sin fecha). *Variables en Python.* https://j2logo.com/python/tutorial/variables-python/ (Julio, 2020).
- Covantec. (Sin fecha). *Variables y constantes.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/variables_constantes.html#variables-y-constantes (Enero, 2021)
- Openbookproject. (2009). 2. *Variables, expresiones y sentencias.* http://www.openbookproject.net/thinkcs/archive/python/thinkcspyesp3e_abandonado/cap02.html (Julio, 2020).

Actividad 6: Mi primer app gráfica con PyGame

<table>
<tr><td colspan="2">Aprendizaje esperado: Ejecutar un programa que muestre el tablero de ajedrez con las 32 piezas e implementar el movimiento individual de los peones.</td><td>Duración de la actividad: 2 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Python 3.x.x
● Anexo MPT1A6_ImágenesPiezas.
● Script tablero.py
● PyGame 2.0.0 dev 7
● Archivo tablero.py</td><td>Evidencia:
1. 32 imágenes de las piezas de ajedrez cargadas.

Producto:
1. Script tablero.py con el código necesario para mover las piezas del tablero.</td><td>Retroalimentación:
● Verificar que las imágenes sean cargadas.

Evaluación:
Tomar como criterio de evaluación el uso de las siguientes funciones:
Para la carga de imágenes:
● pygame.image.load()
● pygame.transform.scale()
Funciones para el posicionamiento de las imágenes:
● blit()
● pygame.MOUSEBUTTONDOWN
● pygame.mouse.get_pos()

Para evaluar la segunda parte de la actividad, el movimiento de la pieza en el tablero deberá ser visible.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Colocando mi alianza.
Sugerencia de tiempo invertido: 1 hora.</td></tr>
</table>

En el juego de ajedrez, como en la vida, es importante crear vínculos que nos apoyen y permitan sentirnos confiadas. Por ello, la participante cargará en su tablero de ajedrez el resto de las piezas faltantes que formarán las alianzas correspondientes para cada equipo. Se le solicitará a la participante seguir las siguientes instrucciones:

1. Utilizará el script *tablero.py* de la actividad 4 parte 4.
2. Completará la función *llenaTablero(pantalla)* para que se carguen las 32 piezas del juego en las casillas que le corresponden a ambos colores:
    - Creará 32 variables con diferente nombre, de tal manera que hagan referencia a la pieza del juego que se carga y el equipo que corresponde, por ejemplo: reinaBlanco, peonNegro.
    - Utilizará cada variable para cargar las imágenes de las piezas del tablero con ayuda de la función *image.load()* de pygame.

Segunda parte: Empecemos a jugar.
Sugerencia de tiempo invertido: 1 hora.

Una vez que se cargaron las piezas faltantes en el tablero, la participante tendrá un primer acercamiento a la experiencia de imprimir movimiento a las piezas. Para ello, utilizará el archivo *tablero.py* y elegirá la imagen de la pieza que prefiera para posteriormente, realizar las siguientes instrucciones:

- Sustituirá los números de la posición de la imagen utilizada en la función *blit()* por dos variables que tengan la posición almacenada.
- Utilizará el evento *pygame.MOUSEBUTTONDOWN* para reconocer cuando se haga click sobre alguna parte de la pantalla.
- Una vez que se reconozca el click, evaluará cuando se presione el botón izquierdo del mouse, y cuando se reconozca este evento, actualizará el valor de las variables para la posición, las coordenadas de la pantalla en las que se encuentra posicionado el cursor con ayuda de la función *get_pos()* de pygame.
- Posteriormente la participante utilizará las variables creadas para determinar la posición de la imagen que eligió.

Notas para apoyar la actividad:

- La participante realizará commit a su script *tablero.py* en su repositorio *juegoDeAlianzas_Ajedrez* de GitHub, con las 32 imágenes cargadas previamente y las modificaciones realizadas en la segunda parte de la actividad.
- Para la primera parte de la actividad:

- Se sugiere utilizar las imágenes que se brindan en la carpeta *ImagenesPiezas*. En caso de que la participante prefiera utilizar otras imágenes, asegurarse de que éstas cuentan con licencia creative commons o que no cuenten con derechos de autor, de no ser así, no podrán ser utilizadas.
- En caso de que las imágenes no cuenten con el tamaño adecuado para cada casilla, la participante deberá escalarlas con ayuda de la función *scale()* de PyGame, que puede consultar en la misma documentación del programa.
- La participante deberá asegurarse de que todas las imágenes cuenten con la misma escala; para ello se sugiere el uso de una variable para la definición del ancho y alto. Será posible definir nuevas variables para el escalamiento de las imágenes; en caso de ser así, declarar las 32 variables necesarias.
- Se asegurará que la participante utilice la función *blit()* para colocar cada pieza sobre su posición correspondiente en el tablero construido en la actividad anterior.

- Para la segunda parte de la actividad:
  - Se sugiere hacer uso de los enlaces para aprender más.
  - Se recomienda como criterio de evaluación que el cambio de posición de la pieza del tablero sea visible.
  - Se recomienda asegurarse que la participante tenga cuidado en la ubicación de las variables en el código, es decir, considerar las variables a nivel global y local.
- Para el desarrollo de esta actividad no es necesaria la compresión total del funcionamiento del ciclo *if*.

Links para aprender más:

- Design&Programing. (2016). pygame tutorial 5~~*Eventos del teclado y mouse/moviendo objetos.* [video] https://www.youtube.com/watch?v=d1i8uLJ4BWM&list=LLJ6XY9uVNbGlp3XuphTRidw&index=2&t=573s (Junio, 2020).
- Pygame. (Sin fecha). *Pygame front Page*. https://www.pygame.org/docs/ (Junio, 2020).

Actividad 7: Variables de más de una dimensión y sus operaciones

<table>
<tr><td colspan="2">Aprendizaje esperado: Almacenar en una lista de tuplas el nombre, el tipo, el bando y la posición de las piezas de ajedrez y realizar modificaciones.</td><td>Duración de la actividad: 5 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Script tablero.py de repositorio de GitHub Juegodealianzas_Ajedrez.</td><td>Evidencia:
1. Cuadro comparativo MPT1A7_ListasTuplasyDiccionarios completado.

Producto:
1. Modificaciones del proyecto tablero.py
2. Proyecto MPT1A7_VariablesConMasDimensiones con los siguientes scripts
o miLista.py
o miTupla.py
o miDiccionario.py</td><td>Retroalimentación:
● Revisar que en el apartado de características del cuadro comparativo, se abarquen por lo menos los siguientes temas: mutabilidad, sintaxis de listas, tuplas y diccionarios, y qué tipos de datos pueden almacenar dentro de estos; la información deberá plasmarse de manera sintética y corresponder con la documentación de Python vigente.

Evaluación:
● En tablero.py se deberán reflejar los cambios solicitados y deberán ejecutarse sin generar algún error.
● En los scripts miLista.py, miTupla.py y miDiccionario.py, se deberán usar métodos para realizar operaciones en listas tuplas y diccionarios.
● Cuidar que la sintaxis en cada script solicitado no genere ningún error en su ejecución y deberán ejecutar lo solicitado en cada caso.</td></tr>
</table>

Desarrollo de la actividad:

Primera parte: Cajas organizadoras de más de una dimensión.

Sugerencia de tiempo invertido: 3 horas.

a participante realizará una búsqueda acerca de las listas, tuplas y diccionarios, abarcando su definición, características y cómo se declaran en *Python.* El resultado de la búsqueda se materializará en el siguiente cuadro comparativo nombrado "MPT1A7_ListasTuplasyDiccionarios".

Lista	*Tupla*	*Diccionario*
Definición: *Características:* *Declaración en Python:*	...	...

Una vez realizada dicha búsqueda, la participante creará un nuevo proyecto en Pycharm con nombre "MPT1A7_VariablesConMasDimensiones". Se recomienda realizar los siguientes scripts, sin embargo, se dejará a criterio incorporar uno o más ejercicios así lo requieran las necesidades de las participantes; la participante deberá averiguar qué son y cómo accionar cada uno de los métodos enunciados en este ejercicio conjuntando búsquedas en internet y práctica.

Un *script* con nombre "miLista.py":

- Declarará una lista que almacene diferentes elementos, podrá ser, por ejemplo, una lista de artículos, la cual deberá contener distintos tipos de variables (enteras, flotantes y cadenas).
- Agregará al final un nuevo elemento con el método *append()*
- Sacará el último elemento con el método *pop()*
- Invertirá la lista con el método *reverse()*
- Ordenará la lista con el método *sort()*

Un *script* con nombre "miTupla.py*":*

- Declarará una tupla que almacene diferentes elementos, podrá ser, por ejemplo, una lista de artículos, la cual deberá contener distintos tipos de variables (enteras, flotantes y cadenas).
- Contará la cantidad de veces que aparece un elemento con el método *count()*
- Buscará un elemento con el método *index()*
- Probará algún método para modificar listas como *append()* o *pop()*, observará qué sucede y por qué; con este ejercicio la participante deberá reconocer que una tupla no puede modificarse por lo que al intentar implementar estos métodos

ocurrirá un error. Redactará las observaciones en el cuadro comparativo previamente realizado.

Un *script* con nombre “miDiccionario.py”*:*

- Declarará un diccionario que almacene diferentes elementos, podrá ser, por ejemplo, una lista de artículos, la cual deberá contener distintos tipos de variables (enteras, flotantes y cadenas), con sus respectivas claves.
- Accederá a un valor mediante su clave.
- Asignará un valor mediante su clave.
- Obtendrá un valor mediante su clave con el método *get()*.
- Removerá un elemento mediante su clave con el método *pop()* y *popitem()*.
- Actualizará el diccionario con *update()*.

Segunda parte: Simplificando mi Proyecto.

Sugerencia de tiempo invertido: 2 horas.

Se le solicitará a la participante realizar las siguientes instrucciones en su proyecto de ajedrez, para lo cual deberá utilizar el *script tablero.py*:

1. En la función *llenaTablero(pantalla)* creará 32 tuplas donde se almacene en cada una de ellas la posición de cada pieza, es decir, columna y renglón en notación algebraica, para ello podrá recurrir como apoyo al *script misPiezasDeAjedrez.py* realizado en la actividad 5. El nombre de cada tupla será alusivo al nombre de la pieza de ajedrez y su bando, por ejemplo: *reinaBlanca, reinaNegra*, etc.
2. Creará una lista y modificará las declaraciones de las tuplas creadas en el punto anterior para que ahora esta lista almacene todas las tuplas. El nombre de la lista podrá ser *misPiezas, piezas*, o cualquier nombre que haga referencia a las piezas de ajedrez.
3. Creará un diccionario y modificará la declaración del punto anterior para que ahora éste almacene las posiciones, los nombres, y el bando, de las 32 piezas de Ajedrez.
4. Cambiará el valor de la variable de posición de una pieza y deberá verificar que de acuerdo con ese cambio se actualice su posición en el tablero.
5. Cambiará la variable de nombre de una pieza y deberá verificar que de acuerdo con ese cambio se actualice la imagen de esa pieza.
6. Cambiará la variable del bando de una pieza y deberá verificar que de acuerdo con ese cambio se actualice la imagen de esa pieza.

En caso de no presentar ningún error de sintaxis o lógica al momento de ejecutar el *script* y poder visualizar todos los cambios solicitados en los puntos anteriores, la participante realizará un *commit* al repositorio de github *Juegodealianzas_Ajedrez.*

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar el *script*, se le motivará para que realice una búsqueda en internet sobre los elementos necesarios para encontrar o descubrir las soluciones.
- Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la carpeta de evidencias correspondiente.

Links para aprender más:

- Docs.phyton. (Sin fecha). *Documentación de Python 3.8.6.* https://docs.python.org/es/3.8/ (Julio, 2020).
- CosasDeDevs. (2020). *Cómo usar listas, diccionarios, tuplas y sets en Python.* https://cosasdedevs.com/posts/como-usar-listas-diccionarios-tuplas-y-sets-en-python/ (Julio, 2020).
- Codigazo. (Sin fecha). *Diferencia entre listas, tuplas y diccionarios en Python.* https://www.codigazo.com/python/diferencia-entre-tupla-lista-diccionario-en-python (Julio, 2020).
- COVANTEC. (Sin fecha). *Tipo listas.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo_listas.html (Julio, 2020).
    - *3.10. Tipo tuplas.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo_tuplas.html (Julio, 2020).
    - *Tipo diccionarios.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo_diccionarios.html (Julio, 2020)
- Plascencia, E. (Sin fecha) *Diccionarios en Python.* https://devcode.la/tutoriales/diccionarios-en-python/ (Julio, 2020).

Actividad 8: Control de flujo condicional

<table>
<tr><td colspan="2">Aprendizaje esperado: Construir en bloques condicionales la lógica para mover cada pieza de ajedrez a una posición válida.</td><td>Duración de la actividad: 4 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, cartulina, papel bond, hojas recicladas, papel craft o fichas de trabajo.
● Colores, lápiz, plumas o plumones.
● Python 3.x.x
● Script tablero.py de repositorio de GitHub Juegodealianzas_Ajedrez
● Anexo MPT1A8_Anexos.pdf</td><td>Evidencia:
1. Plantilla resuelta.
2. Script derechosPoliticos.py
3. Script sororidad.py
4. Script derechosSexRepr.py
Productos:
1. Script tablero.py con sentencias para el movimiento de las piezas a través del mouse.</td><td>Retroalimentación:
Para la evaluación de los scripts de la segunda parte de la actividad, deberán tomarse como criterio de evaluación el uso de:
● input()
● Variables
● Uso de if, elif y else.
● Sentencia pass
● Operadores binarios

Evaluación:
El script tablero.py deberá contener ciclos de control (if,elif,else,while) para la evaluación de la posición de las piezas, así como el uso y actualización de las coordenadas, operadores binarios y operadores matemáticos.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: ¿Condiciones para qué?
Sugerencia de tiempo invertido: 1 hora.

La participante realizará una búsqueda en internet sobre los controles de flujo, las sentencias condicionales y los operadores binarios utilizando los enlaces anexos (siguiendo el orden establecido). Deberá buscar los siguientes conceptos:
1. ¿Qué es control de flujo?
2. ¿Cuál es la función de la sentencia if?</td></tr>
</table>

3. ¿Cuál es la función de la sentencia *else*?
4. ¿Cuál es la función de la sentencia *elif*?
5. ¿Qué son los *if* anidados?
6. ¿Qué son y para qué sirven los operadores binarios?

Posteriormente de realizar la búsqueda de información, le corresponderá a la participante elaborar fichas para cada pregunta con el siguiente formato (excepto para los operadores binarios):

¿Cuál es la función de la sentencia _____?	**Diagrama de flujo de la sentencia:**	**Sintaxis:**

Tales fichas le serán de utilidad a la participante para recuperar la información cada vez que lo necesite.
Posterior a la elaboración de las fichas, la participante procederá a resolver el laberinto en la plantilla 1 de los anexos *MPT1A8_Anexos.pdf,* que le conducirá a completar el pseudocódigo que se presenta en la plantilla de 2 de los mismos.

Segunda parte: Mujeres a la obra.
Sugerencia de tiempo invertido: 1 hora.

Para la práctica del control de flujo condicional, se solicitará a la participante la creación de algunos scripts (se recomienda utilizar los descritos a continuación). A ella le corresponderá elaborar la estructura de los mismos, a partir del uso de las sentencias y los operadores binarios que crea adecuados según lo que se solicita.

1. **Script "derechosPoliticos.py".**
   La participante:
   a. Creará una variable que contenga la edad de una mujer.
   b. Elaborará un flujo condicional que muestre el mensaje en pantalla: "A los 18 años las mujeres son consideradas ciudadanas con derecho a votar y ser votadas en elecciones populares. En 1953 se consolidó este logro a partir de

movimientos sociales de mujeres y feministas en México que buscaban el reconocimiento de su participación en la política".

c. El mensaje se mostrará en pantalla sólo si la variable es mayor o igual a 18, si no, se mostrará el siguiente mensaje: "Cuando las mujeres tienen menos de 18 años, igual pueden participar políticamente. Ejemplos de ello son las alianzas que pueden hacer con otras mujeres para hablar de temas que les inquietan y compartir experiencias sobre lo que les pasa en sus vidas. Hoy en día estas alianzas las podemos ver reflejadas en reuniones virtuales para conversar sobre menstruación, su salud sexual y reproductiva, mitos del amor romántico, remedios herbolarios y/o iniciativas en pro de sus derechos humanos dentro y fuera de los partidos políticos."

(Ver apartado de notas).

2. **Script "sororidad.py".**

   La participante:

   a. Preguntará a la usuaria del programa si conoce el significado de "Sororidad" y guardará la respuesta en una variable con ayuda de la función *input()* de *Python.*
   b. Si la respuesta es Sí, mostrará el siguiente mensaje en pantalla: "¡Es genial que conozcas el concepto! Además, los pactos de sororidad o alianzas, sirven para que las mujeres puedan resolver problemáticas en torno a su género (como la negación del acceso a la educación por el hecho de ser mujeres) a partir del reconocimiento mutuo basado en el respeto. La sororidad no es sinónimo de igualdad de pensamiento sólo por ser mujeres, pero sí de apoyo entre mujeres a pesar de las diferencias".
   c. En caso de que la respuesta sea No, mostrará el mensaje: "¿Sabías que? La palabra "sororidad" significa hermandad entre mujeres y sirve para que ellas se identifiquen y pueden aliarse para el logro de un fin común en su beneficio, sin necesidad de tener un pensamiento uniforme en todos los aspectos de su vida."

   (Ver apartado de notas).

3. **Script "derechosSexRepr.py"**.

   La participante:

   a. Preguntará a la usuaria del programa si conoce sus Derechos Sexuales y Reproductivos, y guardará su respuesta en una variable con ayuda de la función *input()*.
   b. Si la respuesta es Sí, preguntará a la usuaria cuántos conoce, y guardará la respuesta en una variable entera.
   c. En caso de que la respuesta sea menor o igual a ocho, mostrará el siguiente mensaje en pantalla: "En México son reconocidos al menos diez Derechos Sexuales y Reproductivos: Derecho a la libertad y autonomía sexual, Derecho a la educación e información sobre sexualidad, Derecho a la salud sexual, Derecho a decidir libremente sobre ejercer o no la reproducción, Derecho a vivir conforme a la propia orientación sexual y a la libre expresión de la misma, Derecho a

vivir conforme a la propia identidad de género y a la libre expresión de la misma, Derecho a la equidad sexual, Derecho a la privacidad y la intimidad, Derecho a procurar el placer sexual y Derecho a la libre asociación sexual".

d. Si la respuesta es mayor a ocho, mostrar el mensaje: "Los derechos sexuales y reproductivos son un derecho humano. En México son reconocidos al menos diez Derechos Sexuales y Reproductivos que en caso de no ser salvaguardados, las leyes mexicanas establecen que se deberán tomar medidas eficaces y oportunas para el ejercicio libre de los mismas. Los Derechos Sexuales y Reproductivos de las mujeres pretenden erradicar la violencia contra las mujeres y el autoritarismo masculino."
e. En caso de que la respuesta a la primera pregunta sea No, mostrar en pantalla el mensaje "Los Derechos Sexuales y Reproductivos son un derecho humano que todas las personas poseen sin importar el género. Estos derechos brindan la posibilidad a las mujeres de decidir cómo controlar todos los aspectos de su salud y en especial el libre ejercicio de tomar decisiones sobre sus cuerpos y sus vidas. En México son reconocidos al menos 10 Derechos Sexuales y Reproductivos". Considerar el caso en el que la respuesta a la primera pregunta sea diferente a sí o no, en ese caso utilizar la palabra reservada que permite no hacer nada en el flujo del programa.

(Ver apartado de notas).

Tercera parte: Construyendo mis reglas.
Sugerencia de tiempo invertido: 2 horas.

La participante procederá a imprimir movimiento a los peones, a través del uso de control de flujo aprendido. Para ello, se le solicitará utilizar el script *tablero.py* para esta parte de la actividad y realizar las siguientes instrucciones:

1. Investigará cómo se mueven los peones a lo largo del tablero, es decir, sus movimientos válidos.
2. Con la información anterior, la participante redactará en un párrafo la explicación del movimiento de los peones y posteriormente intentará representar como le sea posible dicho párrafo en una ecuación matemática.
   (Ver apartado de notas).
3. Complementará la parte dos de la actividad 7 realizando el código correspondiente (haciendo uso de la ecuación anterior) para poder mover las imágenes de los peones en el tablero. Utilizará el control de flujo *if* para corroborar que la pieza se encuentra colocada en una casilla válida del tablero. Deberá considerar que la pieza no puede estar fuera del tablero.
4. Actualizará la posición de las piezas en el diccionario de piezas realizado en la segunda parte de la actividad 8.
5. Utilizará el control de flujo *if* para desarrollar el código que permita evaluar si los peones pueden o no realizar el movimiento en la casilla.

(Ver apartado de notas).

Cuarta parte: ¿Cómo me siento?
Sugerencia de tiempo invertido: 30 minutos.

Se realizará en conjunto con la participante, un diálogo dónde se discutan las siguientes cuestiones:

- ¿Cómo te sentiste con la actividad?
- ¿Qué se te dificultó? ¿Cómo lo solucionaste?
- ¿Ya conocías los datos que se presentaron en cada script de la segunda parte de la actividad?
- ¿Cómo crees que podrías utilizar dicha información en tu vida?

Notas para apoyar la actividad:

- La participante realizará commit a su script *tablero.py* en su repositorio *juegoDeAlianzas_Ajedrez* de GitHub, con las modificaciones realizadas en la tercera parte de la actividad.
- En caso de que los ejemplos de los scripts de la segunda parte de esta actividad se cambien, se asegurará que cuenten con una redacción clara, y que el contenido sea libre de violencia de género (así como de violencia en general), del fomento de estereotipos y del consumo de sustancias tóxicas o dañinas para la salud.

Notas (parte 2):

- En el punto 1. Script *"*derechosPoliticos.py*"* se asegurará que la participante utilice la estructura if-elif.
- En el punto 2 Script *"*sororidad.py**"** se asegurará que la participante utilice la estructura if-else.
- En el punto 3 Script *"*derechosSexRepr.py*"* se asegurará que la participante utilice el esquema *if-elif-else*, *if* anidados y la sentencia *pass*.

Notas (parte 3):

- En el punto 2 se asegurará que en este paso la participante haga uso de: coordenadas (X1, Y1) (X2, Y2), signos matemáticos y operadores binarios para elaborar la ecuación. Se le brindará el apoyo que necesite para realizar la traducción.
- Punto 5 el código correspondiente a las piezas faltantes del tablero se desarrollará en actividades posteriores.

Links para aprender más:

- Javaparajavatos. (Sin fecha). *Movimientos en un tablero de ajedrez*. https://javaparajavatos.wordpress.com/2017/02/18/movimientos-en-un-tablero-de-ajedrez/ (Julio, 2020).
- Pygame. (Sin fecha). *Documentation*. https://www.pygame.org/docs/ (Julio, 2020).
- Bartolomé, M. (Sin fecha). *If… elif… else...* https://www.mclibre.org/consultar/python/lecciones/python-if-else.html (Julio, 2020).
- AprendelA con Ligdi Gonzalez. (2019). *CONDICIONALES ELSE.|# 16 Curso de Introducción a Phyton* https://www.youtube.com/watch?v=FzrXT3ZCbmw (Julio, 2020).

Actividad 9: Control de flujo bucles

<table>
<tr><td colspan="2">Aprendizaje esperado: En varios bucles controlar el flujo del juego de ajedrez.</td><td>Duración de la actividad: 5 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● MPT1A9_Anexos
● Script tablero.py</td><td>Evidencia:
1. Plantilla MPT1A9_Anexos.pdf respondida.
2. Cuadro comparativo MPT1A9_Comparativo.

Producto:
1. Proyecto MPT1A9_Bucles con los scripts de Python.
2. Modificaciones proyecto tablero.py.</td><td>Retroalimentación:
● En plantilla MPT1A9_Anexos.pdf las respuestas a las preguntas deberán tomar en cuenta la función de los bucles, sus características y la documentación vigente de Python.
● Revisar que en el punto 3 de la primera parte, deberán tomar en cuenta las características de los bucles para responder las preguntas.
● El cuadro comparativo deberá describir las características de cada sentencia y la función de cada una.

Evaluación:
● Cuidar que la sintaxis en cada script no genere ningún error en su ejecución y ejecute lo que se está pidiendo en cada script.
● En tablero.py se deberán reflejar los cambios solicitados utilizando los bucles y deberá ejecutarse sin generar algún error.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Una y otra vez.
Sugerencia de tiempo invertido: 1 hora.</td></tr>
</table>

La participantes responderán lo que se pide en la plantilla *MPT1A9_Anexos.pdf* y realizarán búsquedas de información simultáneamente acerca de los bucles *while*, *while...else*, *for* y *for...else*, podrán apoyarse de los enlaces sugeridos en esta actividad; los ejercicios de esta actividad serán recomendados, sin embargo, se dejará a criterio incorporar, modificar o eliminar uno o más ejercicios según lo requieran las necesidades de las participantes.

1. La participante rellenará los recuadros con la información que se le solicita según sea el caso.
2. Posteriormente deberá completar las sentencias para *while* y *for* de acuerdo a la estructura que se le solicita en las plantillas anexas.
3. Para complementar, la participante escribirá la sintaxis de las sentencias *while...else* y *for...else*, y responderá las siguientes preguntas, podrán anotarlas en el reverso del anexo trabajado.
    - ¿En qué se diferencian el bucle *while*, del *while...else*?
    - ¿En qué se diferencian el bucle *for*, del *for...else*?
    - ¿En qué se diferencian los bucles *while* de los *for*?
4. Elaborarán un cuadro comparativo con nombre “MPT1A9_Comparativo”, donde se explique para qué sirven las sentencias *break*, *continue* y *pass* aplicadas en los bucles revisados.

BREAK	*CONTINUE*	*PASS*

5. Los ejercicios realizados en esta parte de la actividad podrán utilizarse como material de consulta en ocasiones futuras.

Segunda parte: Bucles y más bucles.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Como reforzamiento, la participante programará lo siguiente en un nuevo proyecto en pycharm, con nombre “MPT1A9_Bucles”, se recomienda realizar los siguientes ejercicios, sin embargo, se dejará a criterio incorporar más ejercicios así lo requieran las necesidades de las participantes; se motivará a la participante a averiguar cómo resolver los puntos enunciados conjuntando búsquedas en internet y práctica:

- Codificará en *Python* los ejercicios con *for* de la plantilla dos de los anexos *MPT1A9_Anexos.pdf*, esto le servirá a la participante para familiarizarse con el uso de la sentencia, y al mismo tiempo practicará cómo recorrer los elementos

contenidos en las listas, tuplas y diccionarios.

- Usando *while… else*, el *script* solicitará que la participante introduzca una cadena y hasta que ésta no sea igual a la palabra “alianza” no dejará de solicitarla; una vez que sea introducida la palabra correcta, se deberá imprimir un mensaje que diga que la cadena se encontró.
- Usando *for… else*, recorrerá una lista previamente declarada, se podrá usar alguna mencionada en las plantillas de los anexos de esta actividad, y evaluará con un condicional si se encuentra una cadena dentro de la lista, en caso de no encontrarse, se romperá el bucle con la sentencia *break* y se imprimirá un mensaje dentro del *else*.
- Usando *for* y la función *range*, realizará un *script* que pregunte cuántos números se van a introducir, solicite esos números a la participante, y muestre un mensaje cada vez que un número no sea mayor que el primero.
- Adaptará el mismo ejercicio del punto anterior, usando ahora *while*.
- A modo de comentario escribirá dentro del *script* la respuesta a las siguientes preguntas: ¿Qué diferencias observaste al programar un mismo ejercicio con ambos bucles?, ¿Para qué casos crees que sea más adecuado utilizar cada uno de los bucles aprendidos?

Tercera parte: El flujo de mi juego de ajedrez.
Sugerencia de tiempo invertido: 2 horas y 30 minutos.

Se le solicitará a la participante realizar las siguientes instrucciones en su proyecto de ajedrez, para lo cual deberá utilizar el *script tablero.py*:

1. En el ciclo *while* que permite que su programa se mantenga en ejecución, deberá agregar la condición para que en el momento que se detecta la tecla ‘c’ desde su teclado, se pueda salir del juego, se utilizará la sentencia *break.*
2. En la función *llenaTablero(pantalla)* donde se encuentran la estructura de datos que almacena las piezas de ajedrez, deberá utilizar un bucle *while* para cambiar de bando todas las piezas de ajedrez al equipo blancas. Una vez visualizados los cambios, los revertirá para realizar el siguiente punto.
3. Con bucle *while...continue* deberá cambiar de bando solo las piezas del equipo blanco. Una vez visualizados los cambios, los revertirá para realizar el siguiente punto.
4. Con un bucle *while...else* deberá cambiar al bando contrario de cada pieza de ajedrez. Una vez visualizados los cambios, los revertirá para realizar el siguiente punto.
5. Con un bucle *for* deberá contar el número total de piezas, e imprimirá en pantalla cuantas son.
6. Con un bucle *for...else* deberá contar el número total de piezas blancas y el total de piezas negras, e imprimirá el total de

cada una de ellas.

7. En la función *dibujaTablero(pantalla)* reescribirá el código para dibujar su tablero de ajedrez, de forma que ahora utilizará un bucle *while...else...break* para dibujar el tablero completo, es decir dibujará 64 veces una casilla con el bucle.
8. Una vez que se aseguren que con el bucle anterior pueden dibujar el tablero, procederá a reescribir el código pero ahora con un bucle *for* y la función *range* para hacer lo mismo que en el punto anterior.

En caso de no presentar ningún error de sintaxis o lógica al momento de ejecutar el *script* y poder visualizar todos los cambios solicitados en los puntos anteriores, la participante realizará un *commit* al repositorio de github *Juegodealianzas_Ajedrez.*

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar los *scripts*, se le motivará para que realice una búsqueda en internet sobre los elementos necesarios para encontrar o descubrir las soluciones. Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la carpeta de evidencias correspondiente.

Links para aprender más:

- Docs.python. (Sin fecha). *Documentación de Python 3.8.6*. https://docs.python.org/es/3.8/(Julio, 2020).
  - *Sentencia while:* https://docs.python.org/es/3/reference/compound_stmts.html#while (Julio, 2020).
  - *For statement:* https://docs.python.org/es/3/tutorial/controlflow.html#for-statements (Julio, 2020).
  - *La función Range*: https://docs.python.org/es/3/tutorial/controlflow.html#the-range-function (Julio 2020).
  - *Sentencias break, continue, y else en bucles:* https://docs.python.org/es/3/tutorial/controlflow.html#break-and-continue-statements-and-else-clauses-on-loops (Julio, 2020).
  - *Sentencia Pass*: https://docs.python.org/es/3/tutorial/controlflow.html#pass-statements (Julio, 2020).
- Tutorial Python. (Sin fecha). *Bucles while y for en python.* https://www.tutorialpython.com/bucles-while-y-for-en-python/ (Julio, 2020).

- Mc Libre (2019). *Bucle while* https://www.mclibre.org/consultar/python/lecciones/python-while.html (Julio, 2020).
    - *Bucle for* https://www.mclibre.org/consultar/python/lecciones/python-for.html (Julio, 2020).
- Lozano, J. (Sin fecha). *For en Python – El bucle for en Python: estructura y ejemplos* https://j2logo.com/bucle-for-en-python/ (Julio, 2020).
    - *While Python – Bucle while en Python* https://j2logo.com/python/tutorial/python-while-bucle/ (Julio, 2020).
- Programiz (Sin fecha). *Python while Loop* https://www.programiz.com/python-programming/while-loop (Julio, 2020).
    - *Python break and continue* https://www.programiz.com/python-programming/break-continue (Julio, 2020).
    - *Python pass statement* https://www.programiz.com/python-programming/pass-statement (Julio, 2020).

Actividad 10: Funciones

<table>
<tr><td colspan="2">Aprendizaje esperado: Crear funciones para mover las piezas de ajedrez.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, cartulina, papel bond, hojas recicladas o papel craft.
● Colores, lápiz, plumas o plumones.
● Python 3.x.x
● Biblioteca turtle.
● Script funciones.py</td><td>Evidencia:
1. Cuadro sinóptico.
2. Script tortuga.py
3. Script mandala.py

Productos:
1. Script ajedrezFuncional.py</td><td>Retroalimentación:
● El cuadro sinóptico deberá contar con las siguientes categorías: uso de las funciones, sintaxis de una función, argumento de una función, llamar a una función, sentencia return, funciones internas y funciones propias.
● El script tortuga.py deberá contener las funciones triangulo y cuadrado.
● El script mandala.py deberá contener las funciones triangulo y cuadrado, y estar elaborado con ciclos while, for e if para evaluar y dibujar la mandala.

Evaluación:
● El script ajedrez_funcional.py deberá contar con las funciones imprimeTablero(), imprimePieza() y limpiaTablero.()</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Las cosas funcionan mejor.
Sugerencia de tiempo invertido: 1 hora.

La participante revisará un breve script que se le proporcionará con determinadas características, a través del cual reflexionará en un primer momento por qué es necesario el uso de funciones, y posteriormente concretará los conocimientos por medio de una búsqueda en internet.</td></tr>
</table>

1. Se le proporcionará un breve script donde se repitan por lo menos 3 veces una específica sección del código (se recomienda utilizar el script anexo *funciones.py*).
2. La participante deberá identificar qué sección es la que se repite, cuántas veces y qué alternativa se le ocurre para evitar que se repita.
3. A continuación deberá buscar en internet la solución a tal problema.
4. Comunicará la información encontrada, y por medio de un diálogo se discutirá si ésta corresponde al tema de *funciones*. De no ser así, se le compartirá que la solución correcta es a través del uso y creación de éstas.
5. Por último, a través de una búsqueda en internet deberá realizar un cuadro sinóptico (digital o en físico) con las siguientes categorías:
    - Uso de las funciones.
    - Sintaxis de una función.
    - Argumento de una función.
    - Llamar a una función.
    - Sentencia *return*.
    - Funciones internas.
    - Funciones propias.

Segunda parte: Dibujando funcionalmente.
Sugerencia de tiempo invertido: 1 hora.

La participante realizará un script llamado “tortuga.py” e importará la biblioteca turtle al inicio del script; posteriormente se solicitará que realice cada uno de los siguientes puntos:

1. Investigará el funcionamiento y parámetros de las funciones *forward*, *left*, *right* y *exitonclick* de la biblioteca turtle.
2. Utilizará las funciones para dibujar un triángulo.
3. Utilizará las funciones para dibujar un cuadrado.
4. Creará una función llamada *triangulo* que contenga el código del punto dos y mandará a llamar la función para obtener la figura.
5. Creará una función llamada *cuadrado* que contenga el código del punto tres y mandará a llamar la función para obtener la figura.

6. Modificará la función *triangulo* para que reciba como parámetro el tamaño de las líneas que se utilizará para la función *forward* y mandará a llamar la función para mostrar la figura.
7. Modificará la función *cuadrado* para que reciba como parámetro el tamaño de las líneas que se utilizará para la función *forward* y mandará a llamar a la función para mostrar la figura.

Finalizados los pasos anteriores, se solicitará a la participante crear un script llamado “mandala.py” e importará la biblioteca turtle al inicio del script. A continuación llevará a cabo los siguientes puntos:

1. Copiará las funciones *triangulo* y *cuadrado* del script *tortuga.py.*
2. Utilizará ciclos *while, for* e *if* para evaluar parámetros y dibujar una mandala con ayuda de las funciones *triangulo* y *cuadrado* cada vez que una condición se cumpla o no. Realizará combinaciones y pruebas de ciclos hasta lograr obtener una mandala que sea de su agrado.
   (Ver apartado de notas).

Tercera parte: Ajedrez funcional.
Sugerencia de tiempo invertido: 2 horas.

La participante creará un script llamado “ajedrezFuncional.py” y tomará como base para la realización de las siguientes actividades, el script *tablero.py*.

1. Creará una función llamada "imprimeTablero()" que contenga el código del ciclo *for* desarrollado en la actividad 9 para crear las 64 casillas del tablero y desplegarlas en pantallas con ayuda de pygame.
2. Creará una función llamada "imprimePieza()" que reciba como argumento un diccionario (actividad 7) que contenga a su vez todos los datos de las piezas que se quiera imprimir en pantalla y desplegará la imagen en la casilla que le corresponde en el tablero.
3. Creará una función llamada “limpiaTablero()” que permita desplegar el tablero con las piezas en su posición inicial para empezar una nueva partida.

Se asegurará que todas las funciones sean invocadas en una función *main* que contenga el ciclo *while* para permitir desplegar la interfaz de pygame. Para invocar a las funciones deberán desarrollarse ciclos de control que determinen la condición en la que se ejecutará cada función.
(Ver apartado de notas).

Notas para apoyar la actividad:

- Se sugiere que en la parte 1 de la presente actividad, se evite intervenir en la búsqueda de internet que realice la participante.
- En caso de desconocer en qué consiste un cuadro sinóptico, se recomienda revisar la siguiente fuente: Pimienta, J. (2012) *Estrategias de enseñanza-aprendizaje. Docencia universitaria basada en competencias*. México: PEARSON. Tomado de: https://es.slideshare.net/Marangelica2015/pimienta-julio-estrategias-de-enseanza-aprendizaje
- Se recomienda proporcionar a la participante los enlaces anexos para realizar la búsqueda de información.
- En caso de que se decida utilizar un script distinto al anexo para la parte uno de esta actividad, se asegurará de que siga el mismo eje temático de perspectiva de género.

Notas (parte 2):

- Se asegurará que la participante haga uso de al menos dos tipos de ciclos y ciclos anidados, así como de variables.

Notas (parte 3):

- Se sugiere que la participante haga uso de las funciones para lectura del teclado que incluye la biblioteca de pygame, para determinar las condiciones de ejecución de las funciones.

Links para aprender más:

- ApredelA con Ligdi Gonzalez. (2019). *DEFINICIÓN DE UNA FUNCIÓN PARTE 1 | #19 Curso de Introducción a Python.* [video] https://www.youtube.com/watch?v=-1Wcq5Zz4Cl (Agosto, 2020).
- codigofacilito. (2012). *Tutorial Python 12: Funciones.* [video] https://www.youtube.com/watch?v=_C7Uj7O5o_Q (Agosto, 2020).

Actividad 11: Módulos y paquetes

<table>
<tr><td colspan="2">Aprendizaje esperado: Crear e importar un módulo y un paquete.</td><td>Duración de la actividad: 4 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, cartulina, papel bond, hojas recicladas o papel craft.
● Colores, lápiz, plumas o plumones.
● Python 3.x.x
● tablero.py</td><td>Evidencia:
1. Diagrama de árbol.
2. Paquete miPaquete
3. Módulo figuras.py

Productos:
1. Paquete miAjedrez
2. Módulo tablero.py
3. Módulo piezas.py
4. Script juegoAjedrez.py</td><td>Retroalimentación:
● Espacios tienen que ser llenados.
● El paquete miPaquete y módulo figuras.py se debe poder cargar con la sentencia import.

Evaluación:
Para la tercera parte de la actividad, el módulo tablero.py debe contener las funciones imprimeTablero() y limpiaTablero().

La función imprimeTablero() debe contener el código que permita automatizar la creación de los recuadros del tablero con su color correspondiente. Para ello se debe hacer uso de algún ciclo de programación (if, for o while).

La función limpiaTablero() debe permitir reiniciar el tablero y mostrarlo sin piezas.

- El módulo piezas.py debe contener las funciones imprimePieza() y muevePieza().</td></tr>
</table>

		La función *imprimePieza()* debe mostrar todas las piezas del tablero sobre su posición correspondiente. Se debe hacer uso del diccionario que contiene las piezas como un argumento de la función para iterar sobre este y obtener la posición de la pieza. La función *muevePieza()* debe permitir mover una pieza de lugar cuando se detecte un click sobre esta y actualizar su posición en el diccionario que le corresponde a dicha pieza. - El script juegoAjedrez.py debe importar los módulos piezas.py y tablero.py y contendrá una función main() que permita invocar a cada función de los módulos.

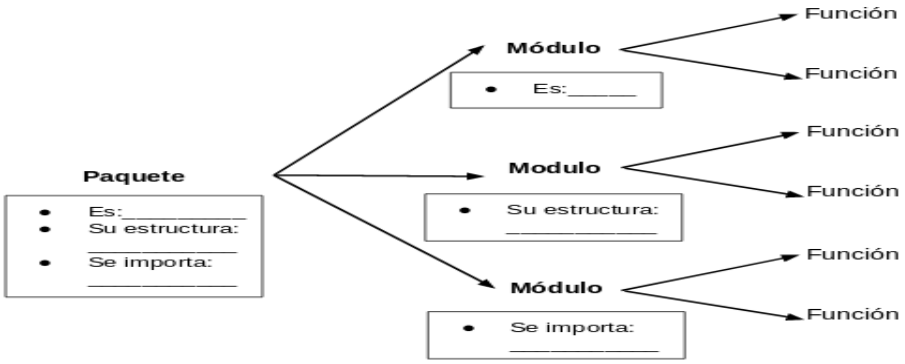
Desarrollo de la actividad:

Primera parte: Modular y empaquetar problemas.
Sugerencia de tiempo invertido: 1 hora.

Se le presentará a la participante la siguiente situación:

- ¿Te ha pasado que tu código tiene un error y te frustras porque no sabes en qué parte específica de todo el código se encuentra el error? Imagina que tienes un script con 15 ciclos, 12 *if anidados*, y 8 funciones, y necesitas revisar línea por línea todo el código para saber qué es lo que está mal. ¿Cómo te sentirías? ¿Puedes imaginar alguna forma para que la revisión del código sea más fácil? ¿Qué se te ocurre?

Estas cuestiones se discutirán por medio de un diálogo; de no ser posible, se recomienda solicitar a la participante que redacte en un pequeño texto sus respuestas a las interrogantes anteriores.
Posteriormente, se le presentará el siguiente diagrama de árbol y se solicitará que complete los espacios vacíos por medio de una búsqueda en internet (se sugiere hacer uso de los enlaces anexos).



Segunda parte: Soy una artista digital.
Sugerencia de tiempo invertido: 1 hora.

Se solicitará a la participante crear una carpeta llamada “miPaquete” y dentro de ella creará un script llamado "figuras.py", el cual servirá como módulo para dibujar figuras en pantalla. En el script deberá realizar las siguientes instrucciones:

1. Del script *tortuga.py* realizado en la actividad 10, copiará la biblioteca *turtle* y las funciones *triangulo* y *cuadrado*.
2. Realizará al menos tres funciones para dibujar tres figuras diferentes a las que ya se tienen. Una figura por función. No deberá incluir la función *exitonclick()* dentro de estas tres.
3. Modificará el script para que aparezca una tortuga como el puntero que dibuja.

Posteriormente, la participante utilizará el script *tortuga.py* para lo siguiente:

1. Importará el módulo *figuras.py* que se encuentra en el paquete *miPaquete.*

2. Realizará una función *main* que contenga un menú de opciones que le permita a la usuaria elegir la figura que desee que la tortuga dibuje.
3. Una vez que la usuaria elija la figura, el script deberá solicitar las dimensiones que se desean para la figura.
4. Por último, la participante utilizará las funciones del módulo para imprimir en pantalla la imagen solicitada.

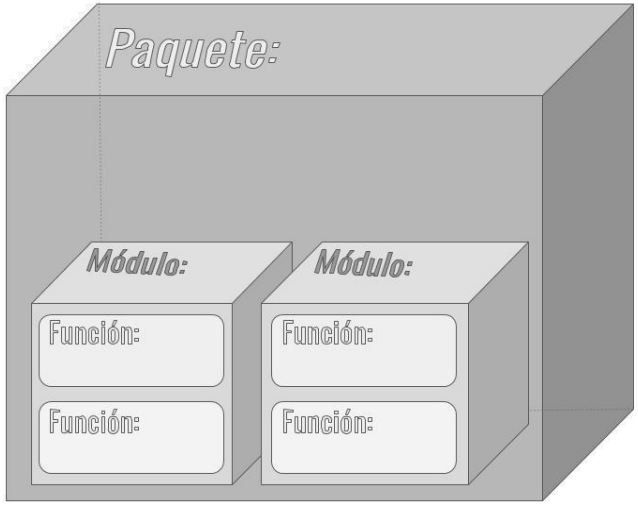
(Ver apartado de notas).

Tercera parte: Ajedrez empaquetado.
Sugerencia de tiempo invertido: 2 horas.

Se solicitará a la participante crear un paquete llamado “miAjedrez” e incluir dentro de él los siguientes módulos:

1. Módulo *tablero.py* que contenga las funciones *imprimeTablero*() y *limpiaTablero()* . La primera función deberá contener el código realizado en la actividad 4 para que se muestren las 64 casillas del tablero de ajedrez.
   La segunda función deberá contener el código necesario para limpiar el tablero una vez que el juego se haya iniciado, se recomienda usar la función *imprimeTablero()* para lo anterior.
   (Ver apartado de notas).
2. Módulo *piezas.py* que contenga las funciones *imprimePieza()* y *muevePieza()* (asegurarse de que la participante copie el código correspondiente para crear estas funciones). La primera función deberá desplegar cada una de las piezas sobre el tablero; se sugiere utilizar el diccionario de cada pieza como argumento de la función.
   La segunda función deberá contener el código que mueva una pieza sobre el tablero; dicho código fue iniciado en la actividad 8 del taller.

Con la estructura anterior de paquete y módulos, se le solicitará a la participante llenar el siguiente diagrama para esclarecer el orden de cada elemento:



Habiendo realizado los pasos anteriores, le corresponderá a la participante realizar un script llamado “juegoAjedrez.py” que importe el paquete y los módulos para hacer uso de las funcionalidades de cada uno. En el script deberá existir una función *main()* que contenga un menú de opciones para elegir cualquiera de las acciones que los módulos proporcionen para el juego. Se evaluará con un ciclo la opción seleccionada por la usuaria e invocará a las funciones en el lugar que les corresponda.

Cuarta parte: ¿Cómo voy?
Sugerencia de tiempo invertido: 30 minutos.

Se realizará un ejercicio de reflexión en conjunto con la o las participantes que cubra los siguientes puntos:

- ¿Qué ventajas identificaste en el uso de funciones, módulos y paquetes?
- ¿Qué es lo que más se te dificultó? ¿Cómo lo solucionaste?
- ¿Cómo te sientes en este punto al ver el avance en tu código?

Como producto final de esta reflexión, se invitará a la participante a compartir su experiencia hasta este momento programando con *Python* a través de un pequeño comentario escrito o impreso en una página de papel, la cual pegará en las instalaciones de PILARES, justo al lado de la infografía que realizaron en la actividad 1.

Notas para apoyar la actividad:

- Para la segunda parte de la actividad, se sugiere que la participante incluya dentro de las figuras a realizar, el símbolo de femenino con la unión de una figura geométrica y dos rectas.
- En caso de que los ejercicios de la parte dos de esta actividad se cambien, asegurarse de que siga el mismo eje temático de perspectiva de género.

Notas (parte 2):

- En este script la participante sí deberá incluir la función *exitonclick()* para el correcto funcionamiento del programa.

Notas (parte 3):

- En punto 1 se sugiere reescribir el código de cada función, haciendo uso de ciclos para su simplificación.

Links para aprender más:

- Pildorasinformaticas. (2017). *Curso de Python. Módulos. Vídeo 34.* [video] https://www.youtube.com/watch?v=t93x-vnFvP4 (Agosto, 2020).
- Tutorial de Python. (Sin fecha) *Módulos y paquetes*. https://tutorial.recursospython.com/modulos-y-paquetes/ (Agosto, 2020).
- Vicungoola Devs. (2017) *12.-Módulos y Paquetes en Python.* [video] https://www.youtube.com/watch?v=8DWLi55Jhv8 (Agosto, 2020).

Actividad 12: Mi primer objeto en Python

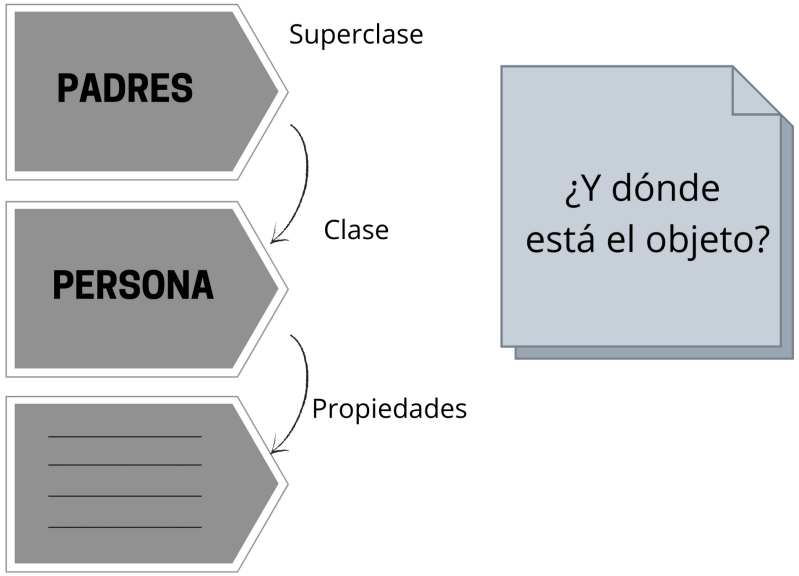
<table>
<tr><td colspan="2">Aprendizaje esperado: Crear un objeto piezaObjeto de ajedrez.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Script tablero.py</td><td>Evidencia:
1. Esquema de la clase Persona con sus respectivas propiedades.

Producto:
1. Script tablero.py modificado con el paradigma orientado a objetos con una clase piezaObjeto y sus propiedades respectivas de la clase, y los 32 objetos de la clase piezaObjeto
2. Script clasepersona.py con la clase Persona.</td><td>Retroalimentación:
● En el esquema de la primera parte de esta actividad, deberá estar contestado el apartado de propiedades; revisar que no se confundan características con acciones.
● Las preguntas en el punto tres de la primera parte de esta actividad deberán estar respondidas con base en el paradigma orientado a objetos.
● Corroborar que se entienda la analogía de la clase persona con su respectiva clase, y propiedades correspondientes.

Evaluación:
● La sintaxis de cada script deberá estar escrita de acuerdo a los criterios del paradigma de la programación orientada a objetos en Python y deberá ejecutarse sin ningún error.
● Deberá ejecutar lo que se pide, deberá contener clase y propiedades y no generar errores en su ejecución.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Mi bienvenida a la POO.</td></tr>
</table>

Sugerencia de tiempo invertido: 2 horas.

1. Las participantes realizarán una búsqueda de información acerca de la importancia de la programación orientada a objetos, clase, objeto, propiedad de un objeto y herencia, podrán apoyarse de los links de apoyo para esta actividad.
2. Se les solicitará que completen el siguiente esquema guiándose de la información que recolectaron, en este caso la clase a trabajar será PERSONA:



- El uso de este ejemplo será para ejemplificar la programación orientada a objetos, en este caso, un ejemplo de las propiedades de la clase *persona*, podrán ser el color de ojos, color de cabello, color de piel, sexo, nombre, etc.; por otro lado, en la viñeta *¿Y dónde está el objeto?* se deberá identificar que hasta que no se inicializa el objeto, es cuando éste adquiere las propiedades de la clase, y es por eso que no aparece integrado en el esquema como tal. Antes de continuar el ejercicio, es importante corroborar que la participante haya comprendido la analogía a estos conceptos.

3. Responderán a las siguientes preguntas: ¿Cuál crees que sea la importancia del paradigma de programación orientada a objetos actualmente?, ¿Qué ventajas y desventajas ofrece este paradigma?; ¿Cómo se construye un objeto en *Python*? y ¿Cómo se elimina un objeto en *Python*? A partir del esquema ¿qué es para ti una clase, un objeto y las propiedades? y ¿qué clase le hereda a qué clase y qué le hereda? Añadirán las respuestas al esquema.

4. Apoyándose en búsquedas en internet, la participante creará un script con nombre “clasePersona.py” y creará la clase Persona, junto con sus atributos (propiedades) que escribieron en el ejemplo, en esta clase deberá representar los atributos como variables y les asignará los valores pertinentes, luego, imprimirá cada una de las propiedades de la clase Persona haciendo uso de la función print. Por ejemplo:

```
objeto = Persona()

print (objeto.propiedad)
```

Segunda Parte: Los moldes para mis Piezas.
Sugerencia de tiempo invertido: 1 hora.

Se le solicitará a la participante realizar lo siguiente en su proyecto de Ajedrez, es decir en su *script tablero.py*

1. Creará una clase de nombre “piezaObjeto”, a esta clase se le añadirán las siguientes propiedades:
    - Bando.
    - Tipo de pieza.
    - Imagen.
    - Posición inicial.
    - Posición actual.

Estas propiedades deberán ir en el método *__init__(self, …)*, hasta el momento la participante no sabrá exactamente qué es un método, por lo que se le pedirá que dentro de la clase cree una función con este nombre y añadirá las propiedades correspondientes.

2. En la función principal de su programa creará 32 objetos de la clase *piezaObjeto,* correspondientes a las 32 piezas de Ajedrez de su tablero, en estas declaraciones pasará como parámetros cada una de las propiedades de cada pieza de la lista enunciada en el primer punto.

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar los *scripts*, se le motivará para que realice una búsqueda en internet sobre los elementos necesarios para encontrar o descubrir las soluciones. Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la carpeta de evidencias correspondiente.

Links para aprender más:

- Documentación de Python 3.8.4. (Sin fecha). *9. Clases.* https://docs.python.org/es/3/tutorial/classes.html (Julio, 2020).
- Vazquez, Monserrat. (2018). *PROGRAMACIÓN ORIENTADA A OBJETOS.* https://www.lainter.edu.mx/blog/2018/03/18/programacion-orientada-a-objetos/ (Julio, 2020).
- Bahit, Eugenia. (Sin fecha). *5.2*. *Programación Orientada a Objetos.* https://uniwebsidad.com/libros/python/capitulo-5/programacion-orientada-a-objetos (Julio, 2020).
- COVANTEC. (Sin fecha). 9.3. *Programación orientada a objetos.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion9/poo.html (Julio, 2020).
- Programacion.net. (Sin fecha). *Cómo funcionan las clases y objetos en Python.* https://programacion.net/articulo/como_funcionan_las_clases_y_objetos_en_python_1505#:~:text=Una%20clase%20es%20un%20tipo,de%20programaci%C3%B3n%20orientado%20a%20objetos. (Julio, 2020).
- if geek then. (2019). *¿Qué es la Herencia en programación orientada a objetos?* https://ifgeekthen.everis.com/es/herencia-en-programacion-orientada-objetos (Julio, 2020).

Actividad 13: Los métodos de mi primer objeto en Python

<table>
<tr><td colspan="2">Aprendizaje esperado: Crear métodos de mostrar, mover y eliminar objeto piezaObjeto.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● script clasePersona.py</td><td>Evidencia:
1. Esquema de la clase Persona con sus respectivos métodos.

Producto:
1. Script tablero.py con tres métodos en la clase piezaObjeto, y 32 piezasObjeto dibujadas en el tablero.
2. Script clasePersona.py modificado, con un método constructor y un método Saludo().</td><td>Retroalimentación:
● En el esquema de la primera parte de esta actividad deberá completarse el recuadro de propiedades, revisar que no se confundan acciones con características.
● Las preguntas en el punto tres de la primera parte de esta actividad deberán estar respondidas con base en el paradigma orientado a objetos en Python.
● Corroborar que se entienda la analogía de la clase persona con su respectiva clase, métodos y propiedades correspondientes.

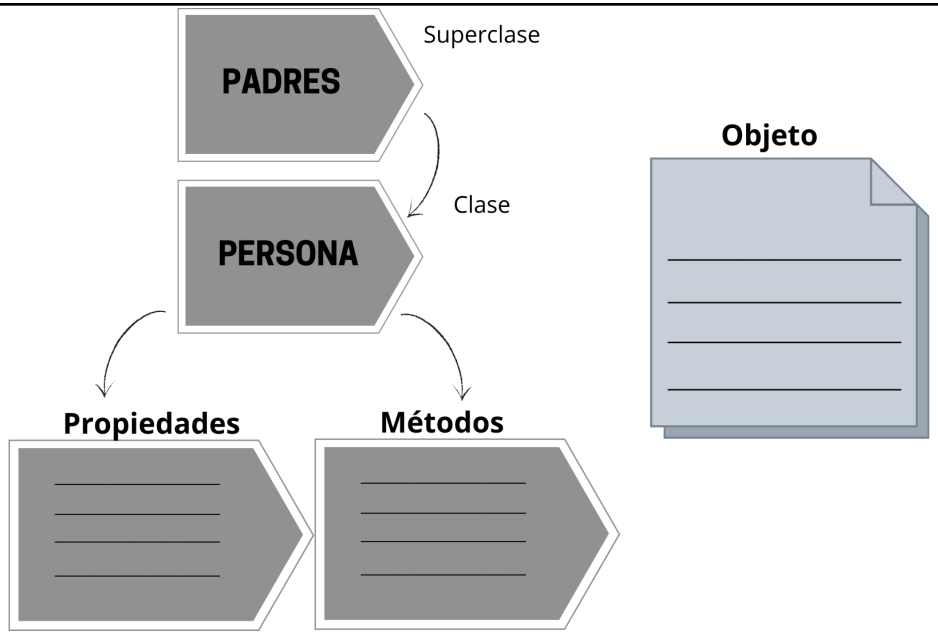
Evaluación:
● La sintaxis de cada script deberá estar escrita de acuerdo a los criterios del paradigma de la programación orientada a objetos en Python y deberá ejecutarse sin ningún error.
● El script clasePersona.py deberá contener una clase Persona y dos métodos, el método constructor para inicializar los objetos y el método saludo() para imprimir un mensaje. No</td></tr>
</table>

		deberá generar errores en su ejecución.

Desarrollo de la actividad:

Primera parte: El comportamiento de mis Objetos.
Sugerencia de tiempo invertido: 2 horas.

1. Siguiendo la dinámica de la actividad anterior, las participantes iniciarán con una búsqueda de información acerca de la importancia de los métodos de un objeto y sus características, podrán apoyarse de los links de apoyo para esta actividad.
2. Retomarán el esquema de la actividad 12 y en este caso añadirán y completarán el recuadro *métodos* apoyándose en las búsquedas realizadas en el punto anterior, un ejemplo de los métodos de la clase *persona*, podrán ser hablar, caminar, pensar, bailar, etc., es decir sus acciones. En el recuadro de objeto, las participantes plasmarán “sus” propiedades y métodos, color de ojos, color de cabello, pasatiempos, etc.



3. Nuevamente apoyándose de búsquedas de información, responderán a las siguientes preguntas:
   ¿Para qué sirve el método __init__() en una clase?
   ¿Por qué el primer parámetro de un método es “self”?
   ¿Cómo realizar métodos en *Python*?
4. La participante utilizará el *script clasePersona.py* realizado en la actividad 12, modificará el código para que ahora en la clase *Persona* se tenga un método constructor para inicializar los objetos de la clase, en este ejercicio, se le presentará a la participante el método constructor como el nacimiento de una persona, si el objeto no es visible hasta que se inicializa, entonces pasaría lo mismo con una persona: no se pueden observar sus características hasta que nacen, así como las acciones que la persona será capaz de realizar, tener en cuenta que las declaraciones de las propiedades de la clase también deberán de ser modificadas, agregando la notación pertinente con la palabra *self,* además deberá agregar un método extra de nombre *Saludo(),* que imprima un mensaje. Al final del script deberá crear un objeto de la clase Persona pasándole los

parámetros correspondientes a las propiedades del objeto que se está inicializando, podrá utilizar el ejemplo que se escribió en la viñeta del objeto.

Segunda parte: Los métodos de mis piezas de Ajedrez.
Sugerencia de tiempo invertido: 1 hora.

Se le solicitará a la participante realizar lo siguiente en su proyecto de Ajedrez, es decir en su script tablero.py

1. En la clase *piezaObjeto* creada en la actividad 12, deberá agregar tres métodos:
    - Un método con nombre *mostrarPieza,* para mostrar la pieza de Ajedrez.
    - Un método con nombre *moverPieza,* para actualizar las posiciones (columna y renglón), de las piezas de Ajedrez.
    - Un método con nombre *eliminaPieza,* para "destruir" la pieza de Ajedrez, el parámetro “activo” se pone en falso y el split se borra.
2. Agregará o modificará el código necesario para dibujar el tablero en pantalla con sus 32 *piezaObjeto*.

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar los scripts, se le motivará para que realice una búsqueda en internet sobre los elementos necesarios para encontrar o descubrir las soluciones. Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la carpeta de evidencias correspondiente.

Links para aprender más:

- Documentación de Python 3.8.4.(Sin fecha). *9. Clases.* https://docs.python.org/es/3/tutorial/classes.html (Julio, 2020).
- Vazquez , Monserrat (2018). *PROGRAMACIÓN ORIENTADA A OBJETOS.* https://www.lainter.edu.mx/blog/2018/03/18/programacion-orientada-a-objetos/ (Julio, 2020).

- Bahit, Eugenia. (Sin fecha). *5.2. Programación Orientada a Objetos.* https://uniwebsidad.com/libros/python/capitulo-5/programacion-orientada-a-objetos (Julio, 2020).
- COVANTEC. (Sin fecha). 9.3. *Programación orientada a objetos.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion9/poo.html (Julio, 2020).
- Programacion.net. (Sin fecha). *Cómo funcionan las clases y objetos en Python.* https://programacion.net/articulo/como_funcionan_las_clases_y_objetos_en_python_1505#:~:text=Una%20clase%20es%20un%20tipo,de%20programaci%C3%B3n%20orientado%20a%20objetos. (Julio, 2020).

Actividad 14: Lógica del juego de ajedrez.

<table>
<tr><td colspan="2">Aprendizaje esperado: Crear un juego de ajedrez en PyGames.</td><td>Duración de la actividad: 6 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Anexo_MPT1A14.pdf</td><td>Evidencia:
1. Realizar una plantilla similar para el movimiento del caballo, alfil, rey y peón.

Producto:
1. Script tablero.py con los métodos moverReina, moverTorre, captura, jaque y jaqueMate.</td><td>Retroalimentación:
● Verificar que los movimientos del caballo, alfil, rey y peón estén bien descritos según las reglas del ajedrez.

Evaluación:
● Comprobar que la reina y la torre se mueven a la posición solicitada en el tablero y si hay obstáculos mostrar mensaje de “movimiento no permitido”.
● Comprobar que una pieza es capturada por otra eliminando la imagen del tablero y cambiando el parámetro de activo a falso.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Movimiento de piezas del ajedrez.
Sugerencia de tiempo invertido: 2 horas.

2. Observar y analizar los movimientos de la reina y de la torre mostrados en el “Anexo_MPT1A14.pdf”.
3. Realizar una plantilla similar para el movimiento del caballo, alfil, rey y peón.
4. Se le solicitará a la participante que agregue un método de movimiento reina en el script tablero.py con el siguiente comportamiento:
a. Si el movimiento es horizontal o vertical, verificar en cada pieza activa del tablero si está en la misma fila que la reina.
i. Si lo está y la nueva posición es a la derecha, explorar con incrementos en las columnas de la posición de la reina hasta alcanzar la posición de la pieza, si la nueva posición de la reina es mayor no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
ii. Si lo está y la nueva posición es a la izquierda, explorar con decrementos en las columnas de la posición de la</td></tr>
</table>

reina hasta alcanzar la posición de la pieza, si la nueva posición de la reina es menor, no se puede mover la reina. De lo contrario mover la reina a su nueva posición.

b. Si el movimiento es hacia arriba o hacia abajo, verificar en cada pieza activa del tablero si está en la misma columna que la reina.
    i. Si lo está y la nueva posición es hacia arriba, explorar con incrementos en las filas de la posición de la reina hasta alcanzar la posición de la pieza, si la nueva posición de la reina es mayor no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
    ii. Si lo está y la nueva posición es hacia abajo, explorar con decrementos en las filas de la posición de la reina hasta alcanzar la posición de la pieza, si la nueva posición de la reina es menor no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
c. Si el movimiento es en diagonal, verificar si la posición de cada pieza activa del tablero coincide con el siguiente criterio si la reina se mueve hacia:
    i. Arriba-izquierda, incrementar las filas en uno y decrementar las columnas en uno, si hay una pieza y la nueva posición de la reina es mayor en las filas y menor en las columnas, no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
    ii. Arriba-derecha, incrementar las filas en uno e incrementar las columnas en uno, si hay una pieza y la nueva posición de la reina es mayor en las filas y mayor en las columnas, no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
    iii. Abajo-izquierda, decrementar las filas en uno y decrementar las columnas en uno, si hay una pieza y la nueva posición de la reina es menor en las filas y menor en las columnas, no se puede mover la reina. De lo contrario mover la reina a su nueva posición.
    iv. Abajo-derecha, decrementar las filas en uno e incrementar las columnas en uno, si hay una pieza y la nueva posición de la reina es menor en las filas y mayor en las columnas, no se puede mover la reina. De lo contrario mover la reina a su nueva posición.

5. Se le solicitará a la participante que agregue un método de movimiento de la torre basado en el comportamiento del número anterior en el script *tablero.py*:
6. Se solicitará a las participantes que en equipos describan en papel el procedimiento de movimiento para caballo, alfil, rey y peón.

Segunda parte: Capturar una pieza contraria en el ajedrez.
Sugerencia de tiempo invertido: 2 horas.

Se le solicitará a la participante realizar lo siguiente en su proyecto de Ajedrez, es decir en su script *tablero.py*

1. Investigar el procedimiento de captura en el ajedrez de la reina y la torre.
2. Se le solicitará a la participante que agregue un método en el script *tablero.py* que a partir del script realizado en la primera parte de esta actividad se elimine una pieza capturada por parte de la reina y de la torre. La eliminación consistirá en borrar visualmente del tablero la pieza y se colocará la pieza que captura en su lugar. Se modificará el parámetro de piezaObjeto que indica que ya no está activa.
3. Responder a las preguntas ¿Qué se necesitaría para que todas las piezas restantes puedan capturar? ¿Qué pasa cuando un rey es capturado?

Tercera parte: Jaque y jaque mate en el ajedrez.
Sugerencia de tiempo invertido: 2 horas.

1. Observar y analizar cuando el rey está en posición de jaque y cuando es jaque mate mostrados en el "*Anexo_MPT1A14.pdf*".
2. Se le solicitará a la participante que agregue un método en el script *tablero.py* para determinar cuando un rey está en jaque y cuando está en jaque mate.

Notas para apoyar la actividad:

- En caso de que la participante tenga dudas acerca de cómo elaborar los scripts, se le motivará para que realice una búsqueda en internet sobre los elementos necesarios para encontrar o descubrir las soluciones. Se procurará no intervenir en la resolución de los problemas, así como en la búsqueda de información.
- Las evidencias generadas en esta actividad podrán elaborarse con cualquier medio y posteriormente, digitalizarse en la carpeta de evidencias correspondiente.

Links de apoyo:

- Bahit, Eugenia. (Sin fecha). *5.2. Programación Orientada a Objetos.*

https://uniwebsidad.com/libros/python/capitulo-5/programacion-orientada-a-objetos (Julio, 2020).

- COVANTEC. (Sin fecha). *9.3. Programación orientada a objetos.* https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion9/poo.html (Julio, 2020).
- Programacion.net. (Sin fecha). *Cómo funcionan las clases y objetos en Python.*
- https://programacion.net/articulo/como_funcionan_las_clases_y_objetos_en_python_1505#:~:text=Una%20clase%20es%20un%20tipo,de%20programaci%C3%B3n%20orientado%20a%20objetos. (Julio, 2020).
- Python. (Sin fecha) *9. Clases.* https://docs.python.org/es/3/tutorial/classes.html (Julio, 2020).
- Vazquez , Monserrat (2018). *PROGRAMACIÓN ORIENTADA A OBJETOS.* https://www.lainter.edu.mx/blog/2018/03/18/programacion-orientada-a-objetos/ (Julio, 2020).

Actividad 15: Mi juego de ajedrez en Python

<table>
<tr><td colspan="2">Aprendizaje esperado: Crear un juego de ajedrez en PyGames.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, cartulina, papel bond, hojas recicladas o papel craft.
● Colores, lápiz, plumas o plumones.
● Python 3.x.x
● PyGame 2.0.0 dev 7
● Documentación de Pygame.
● Tablero.py</td><td>Evidencia:
1. Creación del script botones.py

Productos:
1. tablero.py</td><td>Retroalimentación:
● El script botones.py deberá permitir la manipulación de los botones.

Evaluación:
Para el desarrollo de la primera y segunda parte de la actividad deberá hacerse uso de las siguientes funciones de Pygame:

Para dibujar los botones:
1. draw.rect()
Para mostrar texto sobre los botones:
2. blit()
Para conocer la posición del mouse:
3. mouse.get_pos()
Para reconocer eventos en la interfaz:
4. Pygame.event
Para reconocer el tipo de evento:
5. Event.mode
Para reconocer la pulsación de las teclas:
6. pygame.KEYDOWN
7. pygame.KEYUP</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Sigo informándome.
Sugerencia de tiempo invertido: 1 hora.</td></tr>
</table>

La participante agregará botones a su aplicación. Para ello, se le solicitará seguir las siguientes instrucciones y al mismo tiempo buscar en internet lo que desconozca para el procedimiento correspondiente a creación y reconocimiento de botones, lectura y manejo de teclas, lectura y manejo de mouse, elaboración de menús, y despliegue de textos, todo lo anterior con Pygame.

1. Creará un script llamado “botones.py”, y una ventana gráfica (de las dimensiones que prefiera) con ayuda de un ciclo *while*.
2. Creará dos botones con ayuda de la función *draw.rect()* de Pygame, y les asignará un color diferente a cada uno.
3. Posteriormente, con ayuda de la función *blit()* asignará a los botones los textos: “Reconozco a otras mujeres como iguales a mí” y “Respeto a otras mujeres”.
4. Con ayuda de la función *mouse.get_pos(),* se reconocerá cuando se dé click en alguno de los dos botones.
5. Cuando se reconozca el click sobre los botones, deberá desplegarse en la interfaz dos botones más con los textos “Trabajo con otras mujeres sobre puntos en común” y “Trabajo equitativamente con otras mujeres para lograr un objetivo”, lo anterior haciendo uso de las mismas funciones que se utilizaron en la creación de los dos primeros botones. La evaluación de la posición del mouse la realizará a través del ciclo *if-elif-else* para la ejecución del código correspondiente al botón.
6. Para poder seleccionar los botones que se muestran en el punto anterior, la participante deberá hacerse uso del reconocimiento de teclas con ayuda de *pygame.event*, *event.mode*, *pygame.KEYDOWN*, *pygame.KEYUP* y el reconocimiento de la tecla que se presiona, por ejemplo: *K_LEFT*.
   (Ver apartado de notas).

Al finalizar los pasos anteriores, la participante deberá ejecutar el *script*; en caso de no presentar errores de sintaxis y/o lógica y poder visualizar cada uno de los pasos, deberá continuar con las siguientes actividades.

Segunda parte: Para terminar mi primer proyecto.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante utilizará el script *tablero.py* para realizar las siguientes instrucciones:

1. Creará dentro del flujo principal del programa, tres botones que correspondan a los siguientes elementos: **inicio**, **configuración** y **ayuda**.
2. Utilizará una estructura de control para saber qué botón es el que se selecciona, a través de teclas que la misma participante determine o por reconocimiento de posición del mouse.
3. Cuando se seleccione el botón de **inicio**, deberán mostrarse tres botones que correspondan a los siguientes elementos: iniciar partida, reiniciar partida, salir.

3.1 Para el caso de iniciar partida, esta opción deberá contener el código necesario desarrollado a lo largo de este taller para poder jugar.

3.2 Para el caso de reiniciar partida, esta opción deberá contener el código de *limpiaTablero()* desarrollado durante el taller y permitir iniciar un juego nuevo.

3.3 Para el caso de salir, esta opción deberá contener el código correspondiente a la terminación de la ejecución de la ventana gráfica y por ende del programa.

4. Cuando se seleccione el botón de **configuración** deberá mostrarse un menú de colores que permita a la usuaria elegir el color del tablero. Los colores serán mostrados y elegidos por medio de botones con las combinaciones de colores que la participante prefiera.
5. Cuando se seleccione el botón de **ayuda** deberá desplegarse una ventana en la interfaz que contenga las instrucciones básicas del programa y los créditos de la programadora que realizó el juego. Si se desea, se puede agregar información adicional de relevancia para el proyecto.
   (Ver apartado de notas).

Una vez finalizados los pasos anteriores, en caso de no presentar errores de sintaxis y/o lógica y poder visualizar todos los botones creados, la participante deberá hacer *commit* sobre su repositorio *juegoDeAlianzas_Ajedrez* de Github.

Tercera parte: Soy programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante dará fin a su proyecto de este taller, por lo cual realizará un video de no más de 3 minutos que cumpla con las siguientes características:

- Deberá aparecer la participante en primer plano del video.
- Mostrará imágenes de su juego de ajedrez (en el formato que prefiera: carteles, diapositivas, directamente en la pantalla de la computadora), mientras explica en qué consiste.
- Posteriormente, responderá las preguntas:
  a) ¿Fue una experiencia nueva? ¿Por qué?
  b) ¿Cómo te sentiste a lo largo del taller?
  c) ¿El taller cumplió con tus expectativas?
  d) ¿Recomendarías a otras mujeres que aprendan a programar?
- Como última parte de su video, deberá mostrarse la entrega a la participante del anexo *MPT1A15_Anexos.pdf*.

Este video se hará público en las redes sociales del PILARES que le corresponda, para que pueda estar disponible a todas las personas interesadas.

Notas para apoyar la actividad:

- Para la primera parte, se recomienda que la participante elabore fichas para condensar en ellas la información encontrada durante la realización del ejercicio.
- Para la primera parte, se recomienda que la participante consulte la documentación de Pygame para conocer el comando del reconocimiento de cada tecla del teclado.
- En la segunda parte de la actividad, se deberá organizar y reutilizar todo el código desarrollado a lo largo de este taller para elaborar el código que debe contener los botones. El único código nuevo a desarrollar será el menú de opciones, la elección de colores a través de botones y el desarrollo de la interfaz para mostrar la información de la opción de ayuda.

Notas (parte 1)

- Se sugiere que la participante haga uso de funciones para el desarrollo de esta parte de la actividad, con la finalidad de reforzar lo aprendido y contar con un código entendible.

Notas (parte 2):

- Se sugiere que la participante realice esta parte de la actividad en un *script* diferente a *tablero.py* y desarrolle las instrucciones a través de funciones, con la finalidad de crear un paquete para el tablero y tener el código completo del proyecto dentro de un módulo.

Links para aprender más:

- DaFluffyPotato.(2020). *Menus - Pygame Tutorial.* [video] https://www.youtube.com/watch?v=0RryiSjpJn0 (Noviembre, 2020).
- Codigoplusplus.com.(2015). *Desarrollando Juegos en Python con Pygame - Agregando Un Menu - 13.* [video] https://www.youtube.com/watch?v=PJiFnUywkYc (Noviembre, 2020).

Taller 2: Construyendo una aplicación web con Python.

Este taller se pensó en honor a **Shafrira Goldwasser** (1958 en Nueva York). Es profesora de Ingeniería Eléctrica y Ciencias de la Computación en el Instituto de Tecnología de Massachusetts. Dentro de sus contribuciones, se encuentran sentar las bases de la teoría de la complejidad para la ciencia de la criptografía y pionera en los métodos de verificación en pruebas matemáticas.

Debido a sus investigaciones que plasma a través de artículos, ha recibido varios premios, entre ellos:

- De 1983 hasta 1985, tuvo el Premio IBM para el desarrollo de jóvenes profesores.
- Dos veces el Premio Gödel por el artículo *The knowledge complexity of interactive proof systems* (1993) y *Interactive Proofs and the Hardness of Approximating Cliques* (2001).
- Durante el periodo 2008-2009, obtuvo el Premio Athena Lecturer que se les concede a mujeres destacadas en las Ciencias de la Computación.
- En 2010 ganó la medalla Benjamin Franklin en Ciencias de la Computación, otorgada por el Instituto Franklin.
- En 2017 adquirió el Premio Fundación BBVA, Fronteras del Conocimiento por sus aportes en las Tecnologías de la Información y la Comunicación (TIC).

Módulo: Programación.

Competencia del taller: A través de una plataforma enfocada a la creación de aplicaciones web, la participante desarrollará un sistema para la administración de una biblioteca utilizando el lenguaje de programación Python.

Actividad 1: Las aplicaciones web.

<table>
<tr><td colspan="3">Filosofía E-PILARES: <u>Investigación</u>
En la programación, la habilidad de búsqueda de información es muy importante, porque ayuda a localizar soluciones ya escritas a distintos problemas. Tener presente esta habilidad en el campo laboral, representará una ventaja para librar muchas dificultades. Recuerda: tener actitud de curiosidad y perseverancia para la búsqueda de información.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Identificar las partes más importantes de una aplicación web.</td><td>Duración de la actividad: 2 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Plumas, plumones o lápiz.
● Hojas de papel o papel craft.
● Anexos MPT2A1_Anexos.pdf</td><td>Evidencia:
1. Anexo MPT2A1_Anexos.pdf resuelto.</td><td>Retroalimentación:
● El crucigrama deberá llenarse con las palabras que corresponden a cada concepto especificado en la primera parte de la actividad. En caso de no poder imprimir la plantilla se podrá copiar en hojas de papel o papel craft.
Tanto el crucigrama como la plantilla 5 del anexo, deberán evidenciar el reconocimiento de los diferentes tipos de aplicaciones.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Concepto-grama.
Sugerencia de tiempo invertido: 1 hora.

Se le proporcionará a la participante el anexo MPT2A1_Anexos.pdf. En él se presenta un crucigrama de seis conceptos y las descripciones de cada uno. Se motivará a la participante a realizar una búsqueda de información para completar cada una de las descripciones y de esta manera, adivinar todos los conceptos. A continuación, se enlistan las soluciones:
1. Aplicación que se puede descargar y ejecutar en algún dispositivo móvil como tablet, celular, iPad, etcétera: aplicación móvil.
2. Tipo de aplicación web que muestra el mismo contenido para todas y todos los usuarios; es decir que el único propósito del sitio es brindar información y no cuenta con interactividad con las y los usuarios: estática.
3. Está compuesta por tres partes básicas: un servidor web, una conexión de red y clientes. Algunos tipos que existen son: modelo</td></tr>
</table>

de capas, modelo de dos capas y modelo de tres capas: *arquitectura web.*

4. Aplicación que se encuentra instalada en el sistema operativo de la computadora de la o el usuario. Generalmente está elaborada para desarrollar una tarea en específico, ejemplos: *Power Point*, *Excel*, *Word*, etcétera: *aplicación escritorio.*
5. Aplicación a la que se accede desde una red de internet y a través de un navegador web como *Chrome*, *Firefox*, *Safari*, *Microsoft Edge*, etcétera: *aplicación web.*
6. Tipo de aplicación web que es interactiva con la o el usuario. Se encarga de brindar y capturar información que generalmente se encuentra localizada en una base de datos, cargada y/o actualizada cada vez que la página es visitada: *dinámica.*

Segunda parte: ¿Me quedó claro?
Sugerencia de tiempo invertido: 30 minutos.

Una vez que la participante haya respondido las plantillas 1 a 4 del anexo *MPT2A1_Anexos.pdf*, se le proporcionarán seis ejemplos de aplicaciones, las cuales deberá clasificar en las categorías: aplicación web, de escritorio y móvil. Se recomienda hacer uso de la plantilla 5 del mismo anexo. En caso de que la participante tenga dudas sobre los ejemplos de aplicaciones o desconozca alguna, se le motivará a buscar en internet la respuesta.

Tercera parte: Cliente-servidor.
Sugerencia de tiempo invertido: 1 hora.

Se le presentará a la participante dos casos de la experiencia cotidiana (se sugiere utilizar los que a continuación se enuncian), a través de los cuales identificará las características básicas del modelo cliente-servidor:

1. Caso 1: Te encuentras en un restaurante en el que tu mesa tiene asignado un mesero. Una vez que has leído la carta y decidiste qué ordenar, llamas al mesero para realizar una PETICIÓN de la comida que has elegido (a la que con fines de este ejemplo llamaremos RECURSO). Posteriormente, el mesero comunica tu pedido a la chef y vuelve a tu mesa para retornar una RESPUESTA. Pensemos en dos posibilidades:
    a) El restaurante se ha quedado sin los ingredientes necesarios para preparar tu platillo. En dado caso, la RESPUESTA que recibirás del mesero será negativa, es decir ha ocurrido un ERROR y en consecuencia no podrás obtener tu RECURSO.
    b) La situación contraria es en la que todo ha salido bien y obtendrás una respuesta positiva del mesero, a la que

llamaremos, por ejemplo, OK. De manera que el RECURSO que pediste llegará a ti para que puedas consumirlo.

2. Caso 2: Quieres pedir un artículo de una tienda en línea. Vas a la página y encuentras lo que estás buscando, por lo que decides hacer una PETICIÓN del producto. Consideremos el caso aislado en el que existe una oferta muy llamativa en ese producto que deseas y muchas personas quieren ordenarlo al mismo tiempo. Al presionar click sobre el botón comprar, pueden ocurrir 2 cosas:
    a) Si aún hay artículos en el *stock*, será posible que continúes con el proceso de compra, y como RESPUESTA obtendrás que todo se ha llevado a cabo de manera correcta (es decir un OK), y posterior a unos días de espera, el RECURSO llegará a tu casa.
    b) Si los artículos se han agotado, se te indicará que no hay más productos disponibles, lo que asociaremos con un ERROR, por lo que no será posible que adquieras el RECURSO solicitado.

Una vez que se le presenten a la participante ambos casos, se le solicitará contestar o realizar las siguientes cuestiones:

1. Sin considerar el ámbito tecnológico, ¿cómo definirías a una o a un cliente? ¿has sido una clienta alguna vez? ¿cómo definirías a un servidor?
2. Con los casos anteriores, identifica las partes del modelo cliente-servidor. Puedes apoyarte de las preguntas ¿quién es la o el cliente? ¿quién es el servidor?
3. Dibuja un esquema sobre cómo imaginas el modelo cliente-servidor (asegúrate de incluir los casos de *error* y *ok*). Posteriormente, busca en internet algunos ejemplos de dicho modelo y verifica si tu esquema se parece.
4. Busca y enlista 5 softwares que sirven para montar un servidor local o remoto.
5. Busca y enlista 5 navegadores que sirven para un cliente.

Notas para apoyar la actividad:

- Se recomienda motivar a la participante para que realice una búsqueda en internet sobre la definición de servidor, en caso de que esta no sea clara con la tercera parte de la actividad.

Links para aprender más:

- catarina.udlap.mx/. (Sin fecha). *Capítulo 5. Cliente-Servidor.* http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf (Noviembre, 2020)
- Fazt. (2017). *Modelo Cliente Servidor, Explicación Simple.* [video]

https://www.youtube.com/watch?v=49zdlyLSwhQ (Noviembre, 2020)

- Rodríguez, J. (2018). *Arquitectura Cliente Servidor.* [video] https://www.youtube.com/watch?v=RFAZMtoMzww (Noviembre, 2020)
  profesores.fi-b.unam.mx/. (Sin fecha). *ALGUNAS TECNOLOGÍAS DE LA INFORMÁTICA.* http://profesores.fi-b.unam.mx/heriolg/Infdist9.pdf (Noviembre, 2020)
- Ramo, E. (Sin fecha). *Tópicos avanzados de bases de datos* [imagen] https://eddieramos.files.wordpress.com/2010/02/cliente.png (Noviembre, 2020)
- Sin autor identificado. (Sin fecha). *Aplicaciones cliente - servidor* [imagen] https://lh5.googleusercontent.com/eu17azaTzUG90Eop8vYVZhoxRXkA15FMDMT3qtTlbcKqOqA1dsQLOHn9JcGQjAT1sWs5-_1e87GIe7M2NuDKbu1wpK2Jay2m0OyOvKweEnSkcJM0QNnwKTaassNg-M0rqJRR599bJCeQL7N0 (Noviembre, 2020)

Actividad 2: Diseñando mi primer app web para administrar una biblioteca.

<table>
<tr><td colspan="3">Filosofía E-PIL<u>A</u>RES: <u>Aprender</u> a resolver los problemas:
Es muy común que mientras se está programando, se presenten diferentes dificultades como errores de sintaxis propias del lenguaje en el que se programa, y no siempre se cuenta con el apoyo de una persona que tenga la solución a tales dificultades. Por ello, es de suma importancia aprender a resolver las problemáticas por cuenta propia. Recuerda: la resolución de problemas requiere paciencia, creatividad y constancia.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Analizar y diseñar en papel una aplicación web para administrar una biblioteca.</td><td>Duración de la actividad: 2 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Plumas, plumones o lápices.
● Anexo MPT2A2_Anexos.pdf</td><td>Evidencia:
1. Anexos MPT2A2_Anexos.pdf resueltos.
2. Diagrama de modularización de su aplicación.</td><td>Retroalimentación:
● La sopa de letras de la plantilla 1 de los anexos MPT2A2_Anexos.pdf deberá ser resuelta con cada una de las palabras que se brindan en la misma plantilla.
● En el diagrama de modularización de su aplicación, deberá visualizarse los siguientes módulos: libros, usuarios y préstamos.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Sopa de peticiones.
Sugerencia de tiempo invertido: 1 hora.

Se le proporcionará a la participante el anexo MPT2A2_Anexos.pdf. En la plantilla 1 de dicho anexo, se presenta una sopa de letras con ocho conceptos correspondientes al tema de Protocolo HTTP y métodos de petición. Se le solicitará a la participante resolver dicha sopa de letras con la finalidad de fomentar su lógica, métodos de búsqueda y resolución de problemas, así como la memoria visual. Posteriormente llevará a cabo la búsqueda de cada concepto en internet para complementar su aprendizaje; además deberá realizar notas acerca de lo investigado (se sugiere para lo anterior, utilizar la plantilla 2 y 3 del mismo anexo MPT2A2_Anexos.pdf).</td></tr>
</table>

Segunda parte: ¿Cómo empiezo?
Sugerencia de tiempo invertido: 1 hora.

La participante conocerá las ventajas de la modularidad en la Programación Orientada a Objetos a través de la metodología de estudio de caso, mientras que visualiza y comienza a estructurar el proyecto que desarrollará a lo largo de este Taller 2. Para ello, se le solicitará seguir los siguientes pasos:

1. En las plantillas 4 y 5 del anexo *MPT2A2_Anexos.pdf*, revisará el caso que se presenta sobre una tienda en línea. En tal ejemplo, la participante deberá identificar y enlistar las vistas o los módulos que conforman la página.
2. Posteriormente, apoyándose de la plantilla 6, agregará una pequeña descripción de la función que realiza cada módulo y sus características.
3. Finalmente, complementará el diagrama presentado en la plantilla 7, por medio de escribir el nombre de los módulos del sitio web, dibujar con flechas la forma en que se relacionan entre sí los módulos, y agregar una breve descripción de cada relación.

Una vez concluidos los pasos anteriores, la participante explorará el buscador de por lo menos dos bibliotecas virtuales o digitales (se recomiendan la Biblioteca Virtual Miguel de Cervantes y la Biblioteca Digital UNAM). Realizará la exploración de estos buscadores a través de la búsqueda de algún libro de su interés. A continuación:

1. En la plantilla 8 la participante explicará en un pequeño párrafo lo que observó en la aplicación de la biblioteca, cómo fue el proceso de búsqueda, y qué elementos arrojó la página.
2. Describirá cómo se relaciona la estructura de esta aplicación, con la estructura del ejemplo de la tienda en línea que se le presentó, identificando los posibles módulos con que cuenta el motor de búsqueda de la biblioteca.
3. Posteriormente, en la plantilla 9 enlistará dichos módulos, escribirá las características de cada uno y dibujará flechas para representar cómo se relacionan entre sí los módulos identificados.

Finalmente, se le hará saber a la participante que esos mismos módulos son los que utilizará para la construcción de su aplicación.

Notas para apoyar la actividad:

- Se guiará a la participante durante el procedimiento de la segunda parte de esta actividad, con la intención de asegurar que comprenda en qué consiste la modularidad de la POO.

- Se asegurará que la participante identifique en el caso de la tienda en línea los siguientes módulos: página de inicio, inventario digitalizado (registro de productos), lista de productos, pedido, y lista de pedidos.
- Se asegurará que la participante identifique los módulos: libros, usuarios y préstamos en su aplicación a desarrollar.

Links para apoyar la actividad (parte 2):

- Secretaría de cultura. (2013). *Biblioteca virtual de México.* https://bibliotecavirtualdemexico.cultura.gob.mx/ (Noviembre, 2020).
- Gobierno de México. (Sin fecha). *Biblioteca Vasconcelos.* https://bibliotecavasconcelos.gob.mx/index.php (Noviembre, 2020).
- Universidad Nacional Autónoma de México. (2020). *Biblioteca Digital UNAM.* https://bidi.unam.mx/ (Noviembre, 2020).
- Biblioteca Digital Mundial (Sin fecha). Inicio Biblioteca Digital Mundial. https://www.wdl.org/es/ (Noviembre, 2020).

Actividad 3: Primer paso para construir mi primer app web para administrar una biblioteca.

<table>
<tr><td colspan="3">Filosofía E-PILARES: <u>Sintaxis</u> de un lenguaje de programación.
Cuando aprendemos programación, algunas veces se omite el conocer y memorizar la sintaxis del lenguaje. No obstante, hacer esto puede facilitar la comprensión de todos los elementos posteriores del lenguaje, ya que brinda mayor lógica al aprendizaje. Por ello, asegurarse de tener siempre presente la sintaxis de cada lenguaje que estemos utilizando, puede representar una ventaja durante el desarrollo del código. Recuerda: para memorizar las sintaxis, es necesaria una actitud de disciplina y compromiso.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Desarrollar un método en el framework Flask para generar una página HTML estática de bienvenida.</td><td>Duración de la actividad: 4 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● MPT2A3_Anexos.pdf
● Editor de textos de su preferencia.
● Flask versión 1.1.x</td><td>Evidencia:
1. Código MPT2A3_ConociendoHTML.html funcionando en el navegador.
2. Proyecto “Atenea”, con el script app.py y una carpeta templates que contenga los códigos bienvenida.html y contacto.html

Producto:
1. Plantilla de MPT2A3_Anexos.pdf resuelta.
● Preguntas respondidas sobre frontend y backend.
● Preguntas MPT2A3_preguntas respondidas.</td><td>Retroalimentación:
● Para la plantilla, la descripción de las etiquetas básicas (html, head, body, tittle, meta, link, script, h1, h2 y h3) deberá corresponder con los criterios establecidos por la documentación de HTML vigente.
● Asegurarse de que las participantes identifican la tecnología que están usando en su proyecto, ya sea en el frontend y backend.
● Las respuestas deberán corresponder a lo establecido en la documentación de Python y Flask vigente.

Evaluación:
● El código, MPT2A3_ConociendoHTML.html deberá visualizarse en el navegador sin errores.
● Proyecto funcionando en la URL:</td></tr>
</table>

		http://127.0.0.1:5000/. ● El proyecto deberá realizar lo que se le solicita y no generar ningún error en su ejecución.

Desarrollo de la actividad:

Primera parte: ¿Cómo se escriben las páginas web?

Sugerencia de tiempo invertido: 1 hora.

1. Como primer ejercicio, la participante realizará una búsqueda apoyándose en la plantilla *MPT2A3_Anexos.pdf,* sobre aspectos básicos de *HTML* como su definición y para qué sirve, así como sus etiquetas básicas: *html, head, body, title, meta, link, script, h1, h2 y h3.*

2. La participante realizará lo siguiente en la plantilla:

   A. Rellenará los recuadros con la información que le solicita la plantilla y realizará búsquedas de información simultánea acerca de aspectos básicos de *HTML* y sus etiquetas. Será importante fomentar que la participante logre revisar la información por sí misma en búsquedas de internet.
   B. Escribirá el código incluido en el anexo en un editor de texto, se recomienda usar *Visual Studio Code* o cualquier otro deseado, y guardará el archivo como "MPT2A3_ConociendoHTML.html".
   C. Ejecutará el código en su navegador preferido e identificará las etiquetas que se escribieron en el archivo *MPT2A3_ConociendoHTML.html*, reflejadas en el navegador.

El archivo que contendrá su código y el anexo respondido se subirán a la carpeta de evidencias correspondiente.

Segunda parte: Empiezo a crear mi aplicación.

Sugerencia de tiempo invertido: 1 hora.

La participante realizará una búsqueda de información acerca de las particularidades y generalidades de *Flask*, es decir, sus características, sus ventajas sobre otros *frameworks* y lo que pueden hacer con el mismo; posteriormente, realizarán los siguientes ejercicios:

1. Como primer paso para la creación del proyecto de este taller, la participante deberá instalar *Flask*, por lo que se recomienda crear un entorno virtual en la computadora para seguir con los pasos posteriores. Para ello se le pedirá que se apoye principalmente en la documentación de *Flask*, o utilice otros recursos como videotutoriales o búsquedas en la web. En el proceso de creación del entorno virtual la participante nombrará a la carpeta que contendrá su proyecto como: "Atenea".
2. Una vez creado el entorno virtual, dentro de la carpeta del proyecto creará un *script* de nombre "app.py" donde comenzará a escribir su aplicación. Para ello se le pedirá que mediante una búsqueda en internet responda a las siguientes preguntas, podrá anotar sus respuestas en una hoja de papel o en cualquier editor de texto bajo el nombre "MPT2A3_preguntas".
    - ¿Qué es un decorador en *Python*?
    - ¿Qué es una ruta?
    - ¿Cómo puedo definir una ruta en *Flask*?
3. En el archivo *app.py* escribirá el código *python* para generar un mensaje que imprima el mensaje "¡Bienvenida!" en una página en el navegador. Para ello hará lo siguiente:
    a. Importará la clase *Flask* del módulo *flask*, utilizando las sentencias *from…import* al inicio de su *script*.
    b. Creará una aplicación *Flask*, instanciando el objeto *Flask*.
    c. Definirá una ruta para la raíz, utilizando los decoradores de *Python*, y escribirá una función de nombre "index()" que será el método que retornará el mensaje de bienvenida.
4. Apoyándose nuevamente de la documentación de *Flask* o de cualquier otro recurso la participante buscará los comandos necesarios para ejecutar el código que escribió.
5. Observará el mensaje en el navegador, en la dirección del servidor donde se está ejecutando su proyecto.

Tercera Parte: Escribo mi página web y se la doy a Python.

Sugerencia de tiempo invertido: 2 horas.

La participante modificará su proyecto *Atenea,* trabajado en la parte dos, de esta actividad con los siguientes ejercicios:

1. Escribirá el código *HTML* necesario para construir una página web simple, parecida a la que se construyó en la primera parte de esta actividad. Colocará como título de la página web el nombre de la Biblioteca Virtual que se va a realizar a lo largo del

taller, y dentro de la etiqueta *body* escribirán el siguiente mensaje: “Bienvenida al sistema de administración de la biblioteca Atenea” .

2. En la carpeta principal de su proyecto, creará una carpeta de nombre “templates”, y ahí guardará el código *html* con nombre “bienvenida.html”.
3. La participante deberá realizar una búsqueda para conocer las plantillas de renderizado que utiliza *Flask*, en específico la función *render_template()*. En el archivo *app.py* reescribirá el método *index()* para que en lugar de retornar el mensaje “¡Bienvenida!” retorne una llamada a la función *render_template()* pasándole como parámetro el archivo *html* realizado en el punto número dos.
4. Visualizará los cambios hechos a su proyecto en el navegador.
5. Siguiendo los mismos pasos anteriores, la participante creará una página web que muestre los datos de contacto ficticios de la biblioteca, por ejemplo el nombre, teléfono y correo electrónico, y guardará el archivo con nombre “contacto.html” en la carpeta *templates*.
6. Escribirá una nueva ruta, la de contacto, en el *script app.py* con un método que mande llamar la página descrita en el punto anterior, y la observará en el navegador.

Una vez visualizados los cambios y que el *script* no presente errores en la ejecución, la participante creará un repositorio en *GitHub* para alojar su proyecto, con el nombre “Atenea”, en este hará el primer *commit* para subir los archivos del proyecto.

Cuarta Parte: La parte visible y no visible de mi web.

Sugerencia de tiempo invertido: 30 minutos.

La participante realizará una búsqueda acerca de los conceptos *frontend* y *backend*, y a partir de la información recolectada, responderá las siguientes preguntas, ya sea conversando de participante a participante (si se prestan las condiciones), o de participante a tallerista.

- ¿Con qué tecnología estoy construyendo el *frontend* de mi Biblioteca?
- ¿Con qué tecnología estoy construyendo el *backend* de mi Biblioteca?

Se guiará a la participante para que logre identificar la tecnología que están usando para su proyecto.

Links para aprender más:

- flask.palletsprojects.com. (Sin fecha) *Installation.* https://flask.palletsprojects.com/en/1.1.x/installation/ (Agosto, 2020).
    - *Quickstart*.

  https://flask.palletsprojects.com/en/1.1.x/quickstart/ (Agosto, 2020).

Actividad 4: Mejorando la apariencia de mi app.

<table>
<tr><td colspan="3">Filosofía E-PILA <u>R</u> ES: <u>Repasar</u>
Está comprobado que para adquirir un conocimiento, es indispensable practicarlo un sinnúmero de veces. El repasar una y otra vez lo que se ha aprendido en este módulo, reafirmará los conocimientos, permitiendo que estos permanezcan con mayor facilidad.
<u>Recuerda: la práctica hace a la maestra, por lo que repasar los conocimientos demanda constancia, disciplina y compromiso.</u></td></tr>
<tr><td colspan="2">Aprendizaje esperado: Agregar una biblioteca (Bootstrap) al proyecto para mejorar la apariencia y la funcionalidad de la aplicación.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● MPT2A3_ConociendoHTML.html
● MPT2A4_SimilitudDiferencia</td><td>Evidencia:
1. Maquetado de la página de bienvenida.
2. Archivo index.html.
3. MPT2A3_ConociendoHTML.html modificado.
4. Archivo bienvenida.html modificado.
5. Archivo app.py modificado.
6. Archivo base.html.

Producto:
1. Cuadro MPT2A4_SimilitudDiferencia completado.</td><td>Retroalimentación:
● Corroborar que las participantes identifiquen visualmente que las páginas que son responsivas, se adaptarán al dispositivo en el que se abre la página web y las que, de lo contrario, no cuentan con esta característica, no se adaptarán.

Evaluación:
● El maquetado deberá contener al menos una barra de navegación como cabecera, un pie de página y el contenido para el cuerpo de la página; se permitirá flexibilidad por posibles cambios en el futuro.
● Revisar que el archivo index.html contenga las etiquetas correspondientes a la importación de la biblioteca bootstrap; deberá visualizarse en el navegador sin errores.
● El archivo</td></tr>
</table>

		MPT2A3_ConociendoHTML.html deberá estar modificado con las etiquetas *div*, *span*, *ul*, *li* y *p* añadidas, y las etiquetas *button* y *a*, de uso opcional.

Desarrollo de la actividad:

Primera parte: ¡Más etiquetas...!

Sugerencia de tiempo invertido: 1 hora.

1. La participante realizará una búsqueda de las nuevas etiquetas *HTML*: *div*, *span*, *a*, *ul*, *li*, *button*, y *p.*
2. Para identificar su funcionamiento y practicar las nuevas etiquetas, la participante trabajará nuevamente sobre el código *MPT2A3_ConociendoHTML.html* de la actividad pasada (creada en Actividad 3) en este caso, incorporando las nuevas etiquetas.
    - Modificará de las frases las etiquetas *h3* para que ahora se enlisten con las etiquetas *ul* y *li.*
    - Utilizará dos etiquetas *p* para añadir los siguientes párrafos:

¿Sabías que... hay violencia que vemos a diario, y que es difícil de percibir?
Un ejemplo son las violencias invisibles que son acciones o comentarios sutiles, dirigidos a las mujeres porque se piensa que el ser mujer socialmente significa ser débil o tener menos capacidades para realizar ciertas actividades. Algunos ejemplos se observan en los ***comentarios sexistas*** *que muestran ideas o creencias de lo que puede y debe hacer una mujer, algunas experiencias que ejemplifican esto dicen:*

Susana: "El otro día fui a recoger un paquete a la taquería, y los señores al verme en bicicleta bajo la lluvia, comentaban importandoles poco que yo podía escucharlos: "¿ya viste a la chica, Gonzalo? ¡y tú ni sales a entregar a los comensales si te da frío!, ***¡y eso que es mujer!".*** *Me dejó pensando, y cuestioné... ¿qué tiene que ver el que yo reparta en bicicleta bajo la lluvia, con que sea mujer?"*

- Siguiendo la misma dinámica del punto anterior, escribirá en otro párrafo un breve relato de una acción machista que

haya vivenciado o visto.

- A partir de la información que recolectaron sobre las etiquetas *div* y *span,* incorporarán al código ambos textos (el que se presenta, y el párrafo del punto anterior) de la manera que la participante crea conveniente.
- El uso de las etiquetas *a* y *button* será libre.

Segunda parte: Adaptarse o quedarse en el olvido.

Sugerencia de tiempo invertido: 1 hora.

- Para este ejercicio, se le proporcionará a la participante dos links a distintos sitios web, uno responsivo y otro no responsiva; deberá abrir por lo menos una página de cada sitio en un navegador de computadora y en un dispositivo móvil, ya sea *tablet,* teléfono celular, etcétera; observará la adaptación de las páginas con característica responsiva en diferentes dispositivos y completará el siguiente cuadro con nombre *MPT2A4_SimilitudDiferencia:*

¿Qué diferencias o similitudes observas?	
Página con web responsiva.	Página sin web responsiva.

Antes de pasar al siguiente ejercicio, se corroborará que la participante a través de la observación y por sí misma, llegue a la conclusión de que las páginas que son responsivas se adaptan al dispositivo móvil en el que se abre la página web y las que de lo contrario, no cuentan con este característica, no se adaptarán.

- Una vez finalizado el ejercicio anterior, como reforzamiento de los conocimientos, la participante realizará una búsqueda en internet sobre qué es una web responsiva.
- De acuerdo con la información recolectada, realizará un maquetado de cómo se vería su página principal de bienvenida en computadora y en un dispositivo móvil; podrá hacerlo a mano en papel o algún otro medio deseado. De desconocer en qué consiste el maquetado de una página web, se recomienda que la participante consulte los *Links para aprender más*.
- Posteriormente al maquetado, la participante buscará en internet ¿Qué es *bootstrap*? y responderá la pregunta: ¿Es posible crear una web responsiva con esta herramienta? Se podrá añadir la respuesta a estas preguntas al cuadro realizado en el punto

uno de esta parte de la actividad.

- La participante deberá leer la documentación disponible en la página web de *Bootstrap*, para crear una página web que incluya a esta biblioteca y sus dependencias, por lo que creará un archivo con nombre "index.html"*,* copiará la etiqueta *link* que contiene el *css*, y las etiquetas *script* que contienen los *js* de *bootstrap*. Además, agregará una etiqueta *title* y una etiqueta *h1* que muestren un mensaje de bienvenida. Finalmente, visualizará la página en el navegador.

Tercera Parte: Perfecciono mi Biblioteca.

Sugerencia de tiempo invertido: 2 horas.

La participante llevará a cabo las siguientes instrucciones:

1. Realizará una búsqueda sobre ¿Qué es *Jinja*?, ¿Para qué nos sirve en *Flask*?, ¿Cómo pasar una variable a una plantilla?, ¿Cómo mostrar las variables en una plantilla?, y ¿Cómo crear *url's* dinámicas con *url_for()*?
2. Apoyándose en su búsqueda, modificará el script *app.py* del proyecto *Atenea*, para que en el método *index()* se declare el nombre de la biblioteca y el nombre propio de la participante en variables, y estas se retornen en la función *render_template()* para ser mostradas en la página *bienvenida.html*. En la página *bienvenida.html* deberá modificar el *title* y la etiqueta que contiene el mensaje de bienvenida, para que en estas se reciban las variables declaradas en el script *app.py* con la notación de Jinja.
3. Visualizará los cambios hechos a su proyecto en el navegador.
4. En la carpeta *templates* creará un archivo con nombre *base.html* e incorporará *Boostrap*, incluyendo las etiquetas correspondientes, así como se hizo en el archivo *index.html* de la segunda parte de la actividad, y en éste definirá la cabecera, contenido y pie para todas sus páginas web. La participante podrá apoyarse del maquetado de su página web realizado en la parte número dos de esta actividad, rescatando los elementos enunciados en este punto.
   Para realizar este punto se navegará en la documentación de *Bootstrap*, en la sección *Componentes* para incorporar una barra de navegación simple, que direccione hacia la página de inicio y la de contacto, y un *div* para su contenido. Para mostrar contenido dinámico, en el *div* escribirá el código para extender la plantilla, para ello se deberá apoyar de la sección *Herencia de Plantillas* en la documentación de *Jinja*.
5. Modificará el archivo *bienvenida.html* para que ésta funcione como plantilla hija, por lo que escribirá el código para mostrar el mensaje de bienvenida, extendiendo la página *base.html.* Para lo anterior, la participante se apoyará de la sección "Plantilla Hijo" en la documentación de Jinja.
6. Visualizará los cambios hechos a su proyecto en el navegador.

7. Siguiendo los mismos pasos descritos, la participante modificará el método y la ruta de *contacto* en el *script app.py* para que mande a llamar la página web estática *contacto.html*, pero el contenido de ésta, esté almacenado en un diccionario de nombre: “datos”.

Una vez visualizados los cambios y que no se presenten errores en la ejecución, la participante, hará un *commit* al repositorio de *GitHub* para subir los cambios del proyecto.

Links para apoyar la actividad (parte 2):

- Ejemplo página estática: Meza, I. (2019). *El rol de la IA en la sociedad* (Noviembre, 2020) *https://eticaia_unam.gitlab.io/*
- Ejemplo página dinámica: GOBIERNO DE LA CIUDAD DE MÉXICO. (Sin fecha).Explore - *Datos Abiertos Ciudad de México* (Noviembre, 2020) *https://datos.cdmx.gob.mx/explore/?q=mujeres&sort=modified*

Links para aprender más:

- flask.palletsprojects.com. (Sin fecha). *Flask web development, one drop at a time.* https://flask.palletsprojects.com/en/1.1.x/ (Agosto, 2020).
- getbootstrap.com.(Sin fecha). *Introduction.* https://getbootstrap.com/docs/4.5/getting-started/introduction/ (Agosto, 2020).
    - *Alerts.* https://getbootstrap.com/docs/4.5/components/ (Agosto, 2020).
- Project Links.(Sin fecha).*Jinja* https://jinja.palletsprojects.com/en/2.11.x/ (Agosto, 2020).
- jinja.palletsprojects.com. (Sin fecha). *Template Designer Documentation*. https://jinja.palletsprojects.com/en/2.11.x/templates/#template-inheritance (Agosto, 2020).
- PENSANDO EN WEB. (2012). *3.pasar de PSD a HTML y CSS| Maquetando una web desde cero.*

https://www.pensandoenweb.es/pasar-de-psd-a-html-y-css-maquetando-una-web-desde-cero/

- Kunturweb < Expertos en Diseño web />. (2015). *¿Qué es la maquetación de una página web?* https://kunturweb.com/que-es-la-maquetacion-de-una-pagina-web/

Actividad 5: Método Leer de mi aplicación web.

<table>
<tr><td colspan="3">Filosofía E-PILAR <u>E</u> S: <u>Entusiasmo</u>
La programación no tiene por qué ser aburrida. Ésta nos permite crear cosas inimaginables, que pueden llegar a tener un gran impacto en nuestra sociedad. Por ello, programar siempre con entusiasmo e iniciativa puede hacer la experiencia algo mucho más disfrutable, y motivarnos a crear más. Recuerda: programar con actitud de entusiasmo, iniciativa y seguridad puede facilitar el desarrollo de grandes proyectos.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Agregar un método para que muestre un listado de todos los libros que se tienen en un diccionario de datos llamado “libros”.</td><td>Duración de la actividad: 3 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, papel craft o fichas bibliográficas.
● Pluma, lápiz o plumones.
● Archivo base.html del proyecto Atenea.
● Archivo MPT2_ListaDeLibrosAtenea
● Python v.3.
● Jinja v.1.1.
● GitHub.</td><td>Evidencia:
1. MPT2A5_Anexos.pdf resueltos.
2. Fichas de consulta con la información correspondiente a los conceptos bloque for en Jinja, función url_for() y reglas de variables en Flask.

Productos:
1. Archivos libros_lista.html y libro.html</td><td>Retroalimentación:
● Se verificará que las respuestas a las preguntas de la plantilla 2 de los anexos, evidencian las características generales de las bases de datos y del modelo CRUD.
● La información recopilada en las fichas de consulta acerca de bloque for en Jinja, función url_for() y reglas de variables en Flask, deberá ser suficiente para que la participante pueda realizar la tercera parte de la actividad.

Evaluación:
● El archivo app.py deberá contener los métodos libro() y libros_lista(). Se deberá utilizar la función app.route() para agregar la ruta a estos dos métodos.
● La función libro() deberá recibir el id del libro a través del manejo de variables</td></tr>
</table>

		para rutas y convertirla entero. La ruta deberá definirse así: @app.route('/libros/libro/<int:id>') ● La lista *libros* deberá contener al menos cinco diccionarios con un libro diferente cada uno. ● En el documento *libros_lista.html* deberá hacerse uso de un bloque *for* con la sintaxis de *Jinja* para obtener los diccionarios de la lista *libros.* ● En el archivo *app.py* deberá hacerse uso de la función *render_template()* en los métodos para invocar a los documentos *HTML* de la carpeta *templates*.

Desarrollo de la actividad:

Primera parte: Datos por aquí y por allá.
Sugerencia de tiempo invertido: 1 hora.

1. Se le presentará a la participante la historia inventada de una mujer que haga uso de una base de datos en la vida cotidiana, por ejemplo, una agenda telefónica, un diccionario, una cuenta de registro, listas de clientes, u otros (se recomienda utilizar la plantilla 1 de *MPT2A5_Anexos.pdf*). En dicha historia, deberán insinuarse las características de las bases de datos y la lógica CRUD para gestionar los datos.
2. Una vez que la situación haya sido presentada a la participante, ésta deberá realizar una reflexión a través de contestar algunas preguntas con referencia a la definición y características de las bases de datos, así como la definición del término CRUD (se recomienda utilizar la plantilla 2 de *MPT2A5_Anexos.pdf*). Posteriormente, compartirá sus conclusiones de manera oral.

3. Por último para reafirmar el aprendizaje, se le solicitará a la participante la búsqueda en internet de una definición formal de bases de datos y una de CRUD; ambas podrá escribirlas en fichas de consulta como las que ha realizado en algunas actividades de este módulo.

Segunda parte: ¿Qué necesito?
Sugerencia de tiempo invertido: 30 minutos.

Se solicitará a la participante una búsqueda de los elementos técnicos que requerirá para la tercera parte de esta actividad. Los conceptos a buscar son los siguientes:

- *Bloque for* en *Jinja*.
- Función *url_for()*
- Reglas de variables en *Flask*

La información recabada deberá condensarla en fichas de consulta, las cuales retomará en la siguiente parte de la actividad.

Tercera parte: Leer enriquece.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante tendrá un primer acercamiento al método *read* a través de seguir las siguientes instrucciones:

1. Se le solicitará contestar brevemente con sus propias palabras (en tres renglones o menos) la siguiente pregunta: ¿A qué corresponde el método leer o *read* del CRUD?
2. A continuación, se recuperarán de tres a cinco títulos de libros del archivo *MPT2_ListaDeLibros*. Se le presentarán a la participante estos títulos, y se le solicitará buscarlos en internet para identificar los siguientes datos de cada libro:
    - Autora.
    - Año.
    - Editorial.
    - Páginas totales.
    - País.
    - Breve reseña del libro.

3. Una vez identificados los datos, la participante continuará trabajando sobre su proyecto *Atenea* creado en la actividad 3 de este taller realizando los siguientes pasos:
    1. En el archivo *app.py* creará un diccionario por cada libro investigado y almacenará los datos recopilados. La participante se asegurará de que las claves correspondientes a cada tipo de dato (autora, editorial, país, etcétera), sean representativas para el valor que se va a almacenar, por ejemplo: {“anio” : “2017”, “pais”: “México” }.
    2. Posteriormente, guardará los cinco diccionarios creados en una lista llamada “libros”.
    3. Para mostrar todos los libros de la lista en una vista de la aplicación web, la participante creará un documento *HTML* llamado “libros_lista.html” dentro de la carpeta *templates*, y dentro de éste, extenderá la página *base.html*. Este proceso es parecido al realizado en el punto 5 de la parte 3 de la actividad 4 de este taller.
    4. En este mismo archivo (*libros_lista.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:

        - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*
        - Con la sintaxis de un bloque *for* en *Jinja,* recorrerá los diccionarios que se encuentran en la lista *libros*.
        - Por medio de etiquetas *HTML*, mostrará los datos de cada libro. Para lo anterior, recorrerá los elementos de cada diccionario obtenido con el bloque *for*, haciendo uso de la clave con la que definió los contenidos del diccionario en el punto 1 de esta parte de la actividad.
        - Por último, utilizará la función *loop.index()* de *Jinja* para realizar la iteración del bucle en la plantilla.

    5. Una vez que se realizaron las acciones anteriores, la participante escribirá en el archivo *app.py* el método *libros_lista()* para que retorne una llamada a la función *render_template()*, pasando como parámetros el archivo *libros_lista.html*, la lista *libros* que contiene los diccionarios, y la variable que contiene el nombre de la biblioteca.
    6. En seguida escribirá la ruta “/libros/lista” para visualizar lo hecho en los pasos anteriores.
    7. Por último, la participante listará los libros del diccionario uno por uno en otra vista de la aplicación web, para lo cual creará dentro de la carpeta *templates* un documento *HTML* llamado “libro.html” , y dentro de éste extenderá la página *base.html*. Posteriormente en el mismo archivo (*libro.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:

        - Mostrará un mensaje con el nombre de la biblioteca el cual será obtenido del archivo *app.py*
        - Agregará una etiqueta que diga: “Detalles del libro”.

- Listará los datos recopilados de cada libro haciendo uso de la llave definida para cada valor. No será necesario realizar un bloque *for* para obtener el libro, debido a que este será pasado como argumento desde el archivo *app.py*.

8. A continuación en el archivo *app.py,* escribirá el método *libro(id)* que recibirá como argumento el identificador del libro que se desee mostrar. El método deberá retornar una llamada a la función r*ender_template()* pasando como parámetros el archivo *libro.html*, la lista *libros* en el *id* del libro que se desee mostrar, y la variable que contiene el nombre de la biblioteca.
9. Finalmente, la participante escribirá la ruta "/libros/libro/<id>" que recibirá el *id* del libro que se quiera mostrar (el *id* deberá ser convertido a una variable entera). Para lo anterior, será necesario utilizar las reglas de variables de una URL propias de *Flask*. (Ver apartado de notas).
10. La participante guardará todos los cambios, y hará *commit* sobre su repositorio *Atenea* de *Github* en caso de cumplir con lo siguiente: que la lista de libros pueda ser observada en la ruta especificada, que se pueda visualizar cada libro utilizando la ruta con el identificador, y que no se presenten errores durante la ejecución.

4. A manera de reflexión, se le solicitará a la participante retomar la pregunta contestada al inicio de la tercera parte de la actividad. La leerá y deberá complementarla con su respuesta a la siguiente pregunta: ¿Cómo relacionas el método leer o *read* del CRUD, con lo que realizaste en la tercera parte de esta actividad?

Cuarta parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante creará su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla *MPT2_DiarioDeUnaProgramadora* de los anexos):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Notas para apoyar la actividad:

- Se recomienda motivar a la participante a leer alguno de los libros utilizados en la tercera parte de la actividad.

Notas (parte 3):

- Punto 9, el *id* de la *URL* será el que se pase como argumento en el método *libro(id).*

Links para apoyar la actividad:

- Project Links (Sin fecha). Jinja. https://jinja.palletsprojects.com/en/2.11.x/ (Noviembre, 2020).
- flask.palletsprojects.com. (Sin fecha). *Quickstart* https://flask.palletsprojects.com/en/1.1.x/quickstart/ (Noviembre, 2020).

Actividad 6: Método Crear de mi aplicación web.

<table>
<tr><td colspan="2">Aprendizaje esperado: Agregar un método para que cree un registro nuevo en el diccionario de datos llamado “libros”.</td><td>Duración de la actividad: 3 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>

Hojas de papel, papel craft o fichas bibliográficas.
Pluma, lápiz o plumones.
Archivo base.html de proyecto Atenea.
Archivo MPT2_ListaDeLibrosAtenea
Python v.3.
Flask v.1.1
GitHub.

</td><td>
Evidencia:

Fichas de consulta con la definición de las funciones flash(), redirect(), request(), get_flashed_messages(), app.config(), y parámetro Secret Key para la función app.config(); así como los parámetros que reciben cada una.

Productos:

Documento HTML
agregar_libro.html dentro de la carpeta templates del proyecto Atenea.
Script app.py con el método agregar_libro().

</td><td>
Retroalimentación:

Las fichas de consulta deben contener aspectos de las funciones fhash(), redirect(), request(), get_flashed_messages(), app.confing(), y parámetro Secret Key para la función app.config(), como su definición, la funcionalidad y los parámetros descritos.

Evaluación:

Para el archivo agregar_libro.html deberá hacerse uso del archivo base.html para integrar los elementos de la cabecera y estilo.
El formulario en el archivo agregar_libro.html debe desarrollarse con plantillas para formularios incluidas en la documentación de Bootstrap. Puede modificarse el estilo pero la base debe ser tomada de dicho framework.
El tipo de petición del archivo agregar_libro.html debe ser POST.
Se realizarán request en el archivo agregar_libro.html para el envío de datos a otros scripts.

</td></tr>
</table>

		● En el archivo app.py deberán agregar las siguientes instrucciones: -Creación de la ruta "/libros/agregar" con los métodos POST y GET. -Verifica que exista un request desde la página agregar_libro.html para el título del libro. - Agregar los libros al diccionario a través de la función *append()*. -Redireccionar la vista a *libros_lista*.

Desarrollo de la actividad:

Primera parte: ¿Qué necesito?
Sugerencia de tiempo invertido: 1 hora.

Se solicitará a la participante realizar una búsqueda de los elementos técnicos que requerirá para la segunda parte de esta actividad. Los conceptos de *Flask* y *Jinja* a buscar son los siguientes:

- Función *flash()*
- Función *redirect()*
- Función *request()*
- Función *get_flashed_messages()*
- Función *app.config()*
- Parámetro *Secret Key* para la función *app.config()*

De cada función identificará su funcionalidad y los parámetros que recibe; de igual manera agregará una breve explicación sobre cada parámetro.
La información recabada deberá condensarla en fichas de consulta (siguiendo el formato que ha utilizado en alguna actividades de este módulo), las cuales retomará en la siguiente parte de la actividad.

Segunda parte: Las mujeres también escribimos.
Sugerencia de tiempo invertido: 2 horas.
La participante tendrá acercamiento al método *create* a través de construir un formulario que permita agregar libros al diccionario con el uso de una página web. Para ello, se le solicitará seguir las siguientes instrucciones:

1. Se le solicitará contestar brevemente con sus propias palabras (en 3 renglones o menos) la siguiente pregunta: ¿A qué corresponde el método crear o *create* del CRUD?
2. A continuación, seguirá trabajando sobre su proyecto *Atenea* creado en la actividad 3 de este taller realizando los siguientes pasos:
    1. Dentro de la carpeta *templates* creará un documento HTML llamado "agregar_libro.html", y dentro de este, extenderá la página *base.html*. Este proceso es parecido al realizado en la tercera parte de la actividad 4.
    2. En este mismo archivo (*agregar_libro.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
        - Agregará una etiqueta *h* para mostrar el siguiente texto: "Agregar libro".
        - Buscará en la documentación de *Bootstrap* plantillas para formularios, y elegirá la que más se adecue a la petición de los datos para el registro de un nuevo libro a la lista *libros* (Autora, título, año, editorial, páginas totales, país y reseña).
        - Utilizará el método *POST* para el formulario.
        - Utilizará la función *request.form()* en el atributo *value* de la etiqueta *input*, que permitirá capturar los datos solicitados, con la finalidad de enviar los datos del nuevo libro a la aplicación.
    3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "agregar_libro()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *agregar_libro.html* y la variable que contiene el nombre de la biblioteca. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
        - Escribirá la ruta "/libros/agregar" con la opción de métodos *GET* y *POST*.
        - Verificará que el método con el que se ha accedido a la vista es el método POST.
        - De cumplirse la condición anterior, verificará que exista un *request* para el título.
        - De no ser así, mostrará un mensaje con la función *flash()* que mencione que el título es un dato necesario, ya que sin este no tendría sentido el almacenamiento de un libro.
        - De existir el *request* para el libro, creará un diccionario en el que se almacenen los datos obtenidos del formulario, a través de la función *request()* y el campo que se desea guardar en cada una de las llaves del diccionario, por ejemplo: "titulo" : *request.form*('titulo').

- Una vez que se hayan agregado al diccionario todos los elementos del formulario, utilizará la función *append()* para agregar el diccionario a la lista *libros.*
- Con ayuda de la función *redirect()* y *url_for()*, redireccionará a la/el usuario a la vista que muestra todos los libros existentes (*libros_lista*).

4. Finalmente, la participante agregará un libro de la lista anexa en la sección de notas de esta actividad. Recargará la página *libros_lista* y se asegurará de que muestre el libro recién agregado.
5. Si el paso anterior se ejecuta sin presentar problemas, realizará el cambio necesario en el documento *agregar_libro.html* para que en el campo del año del libro, se desplieguen los años desde 1949 al año actual, y se muestren como opciones de selección, con ayuda de la etiqueta *option* de *HTML.* (Ver apartado de notas).
6. Para comprobar que los cambios realizados se ejecutan sin errores, la participante agregará otro libro.
7. Una vez que pueda observarse el formulario, se registren los libros y se puedan visualizar en la lista de libros sin errores, la participante hará *commit* sobre su repositorio *Atenea* de *GitHub*.

3. A manera de reflexión, se le solicitará a la participante retomar la pregunta contestada al inicio de la segunda parte de la actividad. La leerá y deberá complementarla con lo siguiente: ¿Cómo relacionas el método crear o *create* del CRUD, con lo que realizaste en la segunda parte de esta actividad?

Tercera parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla *Diario de una programadora* de los anexos):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Para este momento, se hará en conjunto con la participante un diálogo de retroalimentación, tomando como base todos los diarios elaborados desde la actividad 5 hasta la presente, de tal manera que se discutan las posibles dudas técnicas que aún persistan sobre dichas actividades, así como posibles sugerencias para modificar la dinámica y los ejercicios de las actividades.

Notas para apoyar la actividad:

- Se sugiere motivar a la participante para que investigue y agregue diez libros más a lista de libros por medio del formulario.
- Una lista de títulos se puede encontrar en los anexos “MPT2 Lista de libros”.

Notas (parte 2):

- Punto 5, será necesario crear una variable con la función *range* para obtener los años desde 1949 al año actual, sin necesidad de ingresarlos uno por uno. Utilizará un bloque *for* de Jinja para obtener cada año que establecerá en la etiqueta *option.*

Links para aprender más:

- Muñoz, J. (2018). *flask: Trabajando con peticiones y respuestas (3ra. parte).* https://www.josedomingo.org/pledin/2018/03/flask-trabajando-con-peticiones-y-respuestas-3a-parte/ (Noviembre, 2020).

Actividad 7: Método Actualizar de mi aplicación web.

<table>
<tr><td colspan="2">Aprendizaje esperado: Agregar un método para que actualice un registro en el diccionario de datos llamado libros.</td><td>Duración de la actividad: 2 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Proyecto Atenea.</td><td>Producto:
1. Archivo editar_libro.html, guardado en la carpeta templates.
2. Archivo app.py modificado con el metodo editar_libro().</td><td>Evaluación:
● La plantilla editar_libro.html deberá contener un formulario como el de la plantilla agregar_libro.html, el diseño de esta podrá ser flexible.
● El archivo app.py deberá contener el método editar_libro() Se debe utilizar la función app.route() para agregar la siguiente ruta al método: \libros\editar\<int:id>. El método debe recibir el id del libro a través del manejo de variables para rutas.
● En el archivo app.py debe hacerse uso de la función render_template() en los métodos para invocar a los documentos HTML de la carpeta templates.
● Revisar que la siguiente url no genere errores para editar cada uno de los libros: http://127.0.0.1:5000/editar/<id></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: El futuro es hoy.
Sugerencia de tiempo invertido: 2 horas.</td></tr>
</table>

Se le solicitará a la participante contestar brevemente con sus propias palabras (en 3 renglones o menos) la siguiente pregunta: ¿A qué corresponde el método actualizar o *update* en el CRUD?

Después, realizará lo siguiente en el proyecto *Atenea:*

1. En la carpeta *templates* creará un archivo con nombre "editar_libro.html" y en esta escribirá el código *html* para construir una página web con un formulario como el de la página *agregar_libro.html,* se sugiere se reutilice el mismo código de esta página, ya que el formulario contendrá los mismos campos: título, autor, un menú desplegable de selección para el año, editorial, número de páginas, país y reseña. El botón, en lugar de decir "Agregar" deberá ser cambiado por uno que diga la leyenda "Guardar".
2. En el atributo *value* de la etiqueta *input* deberá complementar el código para mostrar los datos guardados en la solicitud si existe, o de lo contrario, mostrar los datos de la variable *libro* que se pasó a la plantilla que contiene los datos actuales de la base de datos, que es la lista de diccionarios *libros,* en cada uno de los campos. Por ejemplo:

   value = "{{ request.form['autor'] or libro['autor'] }}"

3. En el archivo *app.py* agregará un método de nombre "editar_libro" para editar un libro de la lista de diccionarios *libros.* La ruta para este método será "\libros\editar\<int:id>"
4. Se sugerirá que se reutilice el código escrito en el método *agregar_libro*, haciendo los siguientes cambios para realizar el método que se pide
   a. Se le pasará como parámetro el *id* del libro, ya que con este se identificará el libro a editar, posteriormente, se almacenará en una variable el libro con el *id* que se encuentra en la lista de diccionarios. Como se muestra a continuación:

      libro = libros[id]

   b. En el bloque *if…else…* el diccionario *libro_nuevo,* ahora se llamará *libro_editado,* al final del bloque, se reemplazará el libro con el *id* del libro a editar por la información del *libro_editado,* como se muestra a continuación:

      libros[id] = libro_editado

   c. En la función *render_template()* que se retorna, deberá cambiar la página "*agregar_libro.html*" por la página "*editar_libro.html".*
5. En el navegador, ingresará a la *url* http://127.0.0.1:5000/editar/1 e ingresará nuevos valores a los campos del formulario para editar el libro con *id* 1, una vez se termine de llenar el formulario, dará *click* en el botón "Guardar" y corroborará que éste libro se haya actualizado exitosamente.
6. En la plantilla *libros_lista.html*, agregará un enlace parecido al de *detalles,* en cada registro para que mande llamar al método *editar_libro*. Además, deberá agregar el código necesario para que el campo del año aparezca conforme al valor del registro.

Una vez realizado lo anterior:

- Se le solicitará a la participante que realice las pruebas para corroborar que los libros puedan ser actualizados, por lo que se sugiere que se editen 2 libros de la lista de libros, se reinicie el servidor para observar qué sucede, y con ello responder la siguiente pregunta: ¿Por qué al reiniciar el servidor los cambios no se conservan?
- Luego, se le solicitará que agregue 10 libros y edite los datos de cada uno de ellos.
- La participante hará *commit* sobre su repositorio *Atenea* de GitHub.

A manera de reflexión, se le solicitará a la participante retomar la pregunta contestada al inicio de la primera parte de la actividad. La leerá y deberá complementarla a través de responder: ¿Cómo relacionas el método actualizar o *update* del CRUD, con lo que realizaste en el ejercicio anterior?

Segunda parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.
La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla Diario de una programadora de los anexos):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Será necesario considerar que el diario correspondiente a esta actividad contenga el nombre completo de la participante y el nombre de la actividad en cuestión.

Notas (parte 2):

- Es importante no olvidar que el diario se le solicitará a la participante en cada actividad para ser revisado y tomado en cuenta para las posibles modificaciones del desarrollo de las actividades.

Actividad 8: Método Eliminar de mi aplicación web.

<table>
<tr><td colspan="2">Aprendizaje esperado: Agregar un método para que elimine un registro en el diccionario de datos llamado libros.</td><td>Duración de la actividad: 2 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Proyecto Atenea</td><td>Producto:
1. Archivo app.py modificado con el metodo eliminar_libro().
2. Archivo editar_libro.html con el formulario para el botón de Eliminar.</td><td>Evaluación:
● El archivo app.py deberá contener el método eliminar_libro() Se debe utilizar la función app.route() para agregar la siguiente ruta al método: \libros\eliminar\<int:id>. El método debe recibir el id del libro a través del manejo de variables para rutas.
● En el archivo app.py debe hacerse uso de la función render_template() en el método para invocar a los documentos HTML de la carpeta templates.
● Revisar que el botón de Eliminar que se encuentra en la siguiente url no genere errores para eliminar cada uno de los libros, y se retorne a la página libros_lista.html:
http://127.0.0.1:5000/editar/<id></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: La limpieza es un buen hábito.
Sugerencia de tiempo invertido: 2 horas.</td></tr>
</table>

Se le solicitará a la participante contestar brevemente con sus propias palabras (en 3 renglones o menos) la siguiente pregunta: ¿A qué corresponde el método eliminar o *delete* en el CRUD?

Después, realizará lo siguiente en el proyecto *Atenea:*

1. En el archivo *app.py* agregará un método de nombre "eliminar_libro" para eliminar un libro de la lista de diccionarios *libros.* La ruta para este método será "\libros\eliminar\<int:id>", para la petición *POST.*
2. Se le pasará como parámetro el *id* del libro, ya que con este se identificará el libro a eliminar, luego, dentro del método se sugiere utilizar el método *del* para eliminar el libro con el *id* correspondiente dentro de la lista de diccionarios *libros.*
3. Imprimirá un mensaje con la función *flash()* con la leyenda "El libro X fue exitosamente eliminado" y por último se retornará hacia la página *libros_lista.*
4. En la plantilla *editar_libro*, agregará un formulario con un botón con la leyenda "Eliminar" al final de la página, este formulario deberá llamar al método *elimina_libro* y pasarle como parámetro el *id.* Para realizar esta llamada deberá escribir el código necesario en el atributo *action* del formulario. También se agregará el código necesario para que al dar *click* sobre el botón se muestre un cuadro de confirmación preguntando si se está seguro de eliminar el libro, utilizando el método *confirm* antes de enviar la solicitud.
5. En el navegador, ingresará a la *url* http://127.0.0.1:5000/editar/0 para eliminar el libro con *id* 0, dará *click* en el botón "Eliminar" y observará que este libro se haya borrado exitosamente.

Una vez realizado lo anterior:

- Se le solicitará a la participante que realice las pruebas para corroborar que los libros puedan ser eliminados, por lo que se sugiere que agregue 10 libros y luego elimine cada uno de ellos.
- La participante hará *commit* sobre su repositorio *Atenea* de GitHub.

A manera de reflexión, se le solicitará a la participante retomar la pregunta contestada al inicio de la primera parte de la actividad. La leerá y deberá complementarla respondiendo a la siguiente pregunta: ¿Cómo relacionas el método eliminar o *delete* del CRUD, con lo que realizaste en el ejercicio anterior?

Segunda parte: Diario de una programadora.

Sugerencia de tiempo invertido: 30 minutos.

La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla Diario de una programadora de los anexos):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Será necesario considerar que el diario correspondiente a esta actividad contenga el nombre completo de la participante y el nombre de la actividad en cuestión.
(Ver apartado de notas).

Notas (parte 2) :

- Es importante no olvidar que este diario se le solicitará a la participante en cada actividad para ser revisado y tomado en cuenta para las posibles modificaciones del desarrollo de las actividades.

Actividad 9: ¡Ya sé programar!, pero ¿qué es eso de las bases de datos?

<table>
<tr><td colspan="2">Aprendizaje esperado: Conectar flask con base de datos no estructurada en la nube.</td><td>Duración de la actividad: 5 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Tener una cuenta en un sistema de bases de datos no estructuradas (Mongo Atlas) en la nube.
● MongoDB
● PyMongo
● Proyecto Atenea.
● Anexo MPT2A9_DiagramaEntidadRelacion</td><td>Evidencia:
1. Plantilla MPT2A9_DiagramaEntidadRelacion resuelta.
2. Diagrama Entidad-Relación de la Biblioteca.

Producto:
1. Script “conectar_bd.py”
2. Script “conectar_coleccion.py”
3. Script “agrega_registro.py”
4. Script “agrega_varios_registros.py”
5. Script “lee_registro.py”
6. Script “lee_varios_registros.py”
7. Script “lee_limit.py”
8. Script “lee_query.py”
9. Script “lee_sort.py”
10. Script “edita_registro.py”
11. Script “elimina_registro.py”
12. Script “elimina_registros.py”
13. Script “elimina_coleccion.py”</td><td>Retroalimentación:
● Verificar que se hayan encontrado todas las palabras de la sopa de letras.
● Verificar las relaciones y entidades del diagrama de la base de datos con su funcionalidad en la aplicación.

Evaluación:
● Los scripts deberán correr de forma integrada y operar sobre la base de datos.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: La pista está en la sopa.</td></tr>
</table>

Sugerencia de tiempo invertido: 1 hora.

1. La participante resolverá la plantilla *MPT2A9_LaSopadeLetras*; encontrará cada una de las palabras ocultas que se enlistan y realizará una búsqueda de información de las mismas para responder la plantilla *MPT2A9_SopaResponde* que está a continuación.
2. Una vez resueltos los anexos, la participante realizará una búsqueda de los siguientes métodos y parámetros de MongoDB y los condensará en la siguiente tabla:

	Método/Parámetro	**Definición**
Métodos	*MongoClient()*	
	insert_one()	
	find()	
	sort()	
	limit()	
	delete_one()	
	delete_many()	
	update_one()	
	update_many()	
	drop()	
Parámetros	*insert_id*	

	insert_ids	

Segunda parte: Modelo mi Base de Datos.
Sugerencia de tiempo invertido: 1 hora.

La participante creará un diagrama entidad-relación para entender las relaciones entre las entidades de su biblioteca (se recomienda utilizar el anexo *MPT2A9_DiagramaEntidadRelacion*). Para ello la participante:

1. Enlistará los módulos o entidades que conforman la biblioteca virtual.
2. Responderá las siguientes interrogantes con las cantidades mínimas y máximas solicitadas. Estas deberán estar representadas entre paréntesis, separadas por medio de dos puntos (mínimo:máximo). Ejemplo: (1:10). De tratarse de una cantidad máxima irrepresentable, es posible utilizar la letra m. Ejemplo: (1:m).
    1. Un **libro**, ¿cuántas veces puede solicitarse en un **préstamo**? Respuesta= (min:max)
    2. En un **préstamo**, ¿cuántos **libros** se pueden solicitar? Respuesta= (min:max)
    3. Una **usuaria,** ¿cuántas veces puede solicitar un **préstamo**? Respuesta= (min:max)
    4. Un **préstamo**, ¿A cuántas usuarias pertenece? Respuesta= (min:max)
3. Las preguntas anteriores se agrupan en dos pares, ya que cada par corresponde a una misma relación de entidades. Se indicará a la participante que identifique los dos pares de preguntas, así como las dos entidades que conforman cada relación, y el verbo que las conecta (se recomienda utilizar la plantilla *MPT2A9_DiagramaEntidadRelacion*).
4. Finalmente, en una hoja de papel o detrás del anexo, en caso de imprimirse, la participante realizará lo siguiente:
    1. Dibujará rectángulos y escribirá el nombre de cada una de las entidades dentro de cada rectángulo.
    2. Entre las entidades, dibujará un rombo para representar la relación entre las entidades y escribirá el verbo dentro del rombo.
    3. Escribirá la cardinalidad de las relaciones ayudándose de las respuestas a las preguntas del punto dos.
    4. Escribirá los atributos que conforman cada una de las entidades, y los encerrará en óvalos.

Para este ejercicio, podrán apoyarse de videotutoriales o búsquedas en caso de tener dificultades para realizarlo.
Una vez finalizado el segmento, guardar la plantilla digitalizada en la carpeta de evidencias correspondiente en la nube.

Tercera parte: Mi Base de Datos en la Nube.

Sugerencia de tiempo invertido: 1 hora.

La participante realizará las siguientes instrucciones, para levantar el *cluster* en MongoDB, crear su primer usuario, la base de datos y las colecciones.

1. Creará una cuenta en MongoDB Atlas, en caso de no tener una.
2. Una vez iniciada la sesión, hará click en *Build a Cluster*.
3. Realizará las configuraciones de las secciones *Cloud Provider & Region, Cluster Tier,* y *Additional Settings*.
4. Pondrá un nombre al *cluster* y oprimirá el botón para crearlo.
5. Configurará la seguridad de conexión, en "*Network Access*" y añadirá en "*IP WhiteList*" las *IP's* autorizadas para conectarse al *cluster*.
6. Añadirá el primer usuario Administrador en "*DataBase Access*", seleccionar "*Atlas admin*" y asignará un nombre y password, para este último se podrá hacer uso del autogenerado del password, para tener una contraseña más segura.
7. En el dashboard, creará una base de datos con nombre "biblioteca_bd", y agregará ahí mismo las colecciones de "libros", "usuarias", y "préstamos".
8. Dará click en "*Connect*" y luego en "*Connect Your Application*", seleccionará el controlador *Python* y la versión que están usando.
9. Copiará la cadena que se genera, la cual servirá para conectar la base de datos.

Cuarta parte: Jugando con mi Base de Datos.

Sugerencia de tiempo invertido: 2 horas.

La participante realizará los siguientes *scripts:*

- *Script "*conectar_bd.py*"* :
  En este *script* se escribirá el código para hacer la conexión a la base de datos en la nube, para ello importará la biblioteca *pymongo*, y luego asignará a una variable de nombre *"*cliente*"* la función *pymongo.MongoClient* y se le pasará como parámetro la cadena que se generó en Mongo Atlas en la Tercera Parte de la actividad, no olvidar que a esta cadena se le reemplazará *<password>* y *admin* por los datos del usuario que se creó en la Tercera Parte de la actividad, y *<dbname>* por el nombre de la base de datos. El siguiente paso será obtener la base de datos por lo que se recomienda usar el acceso al

estilo de diccionario, por ejemplo: mi_bd=cliente['biblioteca_bd']. Por último, en un *print* se imprimirá un mensaje que leerá las colecciones creadas en la base de datos, utilizando la función *list.collection_names().*

Si la participante tiene problemas para realizar la conexión se recomienda consultar la documentación de *pymongo* en la sección *"Getting a Database",* el link se proporciona al final de la actividad.

(Ver apartado de notas).

- *Script* "conectar_coleccion.py":

  En este *script* se escribirá el código para revisar que la base de datos *biblioteca_bd* se encuentre en el servidor, para ello, se imprimirá en un *print* la respuesta a una prueba de conexión utilizando la función *test.* Luego, se utilizará la función *list_database_names(),* para obtener una lista de los nombres de todas las bases de datos en el servidor, esta se guardará en una variable de nombre "dblist", y mediante un bloque *if…else* se revisará si *biblioteca_bd* se encuentra en la lista *dblist.* En caso de existir se imprimirá un mensaje de, sí existe esa base de datos, y en caso de no existir, se imprimirá lo contrario.

- *Script* "agrega_registro.py"

  En este *script* se escribirá el código para agregar un registro a la colección *libros* de la base de datos*,* para ello, se obtendrá la base de datos *biblioteca_bd* y la colección *libros,* utilizando el acceso al estilo de diccionario, luego se creará un diccionario de nombre "libro_1", que contendrá los datos de título, autor, año, editorial, páginas totales, portada, país y reseña. Mediante el método *insert_one()* de *pymongo* se insertará en la colección *libros* de la base de datos. Por último, imprimirá en un *print* el *id* del libro insertado, utilizando el parámetro *inserted_id* de *pymongo.*

  (Ver apartado de notas)

- *Script* "agrega_varios_registros.py"

  En este *script* se escribirá el código para agregar más de un registro a la colección *libros* de la base de datos*,* para ello se reutilizará el código del *script* anterior, realizando los siguientes cambios: en lugar del diccionario *libro_1* se creará una lista de diccionarios con nombre *libros,* en esta almacenará al menos tres registros de diferentes libros con los mismos datos que el *script* anterior, para insertar los registros se utilizará el método *insert_many()* y para imprimir los id de los libros *inserted_ids.*

- *Script* "lee_registro.py"

  En este *script* se escribirá el código para leer un registro de la colección *libros* de la base de datos*,* para ello, se utilizará el método *find_one()* para leer el primer libro de la colección y se imprimirán los datos de éste.

- *Script* "lee_varios_registros.py"

En este *script* se escribirá el código para leer todos los registros de la colección *libros* de la base de datos*,* para ello, se utilizará el método *find(),* para leer la colección completa y un bucle *for* para imprimir los datos de cada libro.

- *Script* “lee_limit.py”
  En este *script* se escribirá el código para leer un número limitado de registros de la colección *libros*, para ello, se reutilizará el código del *script* anterior, agregando el método *limit()* al bucle *for* que lee todos los registros de la colección. Como parametro al metodo *limit()* se le pasará el número de registros que se desean leer, por ejemplo el número 2.
- *Script* “lee_sort.py”
  En este *script* se escribirá el código para leer y ordenar bajo un criterio los registros de la colección *libros,* para ello, se reutilizará el código del *script* anterior, cambiando el método *limit()* por el método *sort()* y pasándole como parámetro el criterio, por ejemplo, el año para que los ordene del más antiguo al más reciente.
- *Script* “lee_query.py”
  En este script se escribirá el código para leer los registros que cumplan con un criterio de la colección *libros*, para ello se utilizará el método *find()* que recibirá como parámetro una consulta o query que funcionará como llave para buscar. La consulta se guardará en una variable con nombre "mi_query" y el criterio será a elección de la participante, por ejemplo {anio : 2000}
- *Script* “edita_registro.py”
  En este *script* se escribirá el código para editar un registro de la colección *libros,* para ello se utilizará el método *update_one()* que recibirá dos parámetros, el primero será una consulta o *query* que funcionará como llave para buscar el registro que se va a editar, y el segundo parámetro serán los datos a actualizar del registro, utilizando el operador *$set* de *MongoDB*; se recomienda que se imprima la colección completa antes y después de editar para observar los datos, y que los parámetros se guarden en variables, para el primero en una variable con nombre “mi_query” y para el segundo “nuevos_datos_libros”, luego se le pasarán las variables como parámetros al método *update_one().*
- *Script* “elimina_registro.py”
  En este *script* se escribirá el código para eliminar un registro de la colección *libros,* para ello se utilizará el método *delete_one()* que recibirá como parámetro una consulta o *query* que funcionará como llave para buscar el registro que se va a eliminar; se recomienda que se imprima la colección completa antes y después de eliminar para observar los datos, y que la consulta se guarde en una variable con nombre “mi_query”.
- *Script* “elimina_registros.py”

En este *script* se escribirá el código para eliminar más de un registro de la colección *libros,* para ello se utilizará el método *delete_many()* que recibirá como parámetro una consulta o *query* que funcionará como llave para buscar los registros que se eliminaran; se recomienda que se imprima la colección completa antes y después de eliminar para observar los datos, y que la consulta se guarde en una variables con nombre "mi_query".

- *Script "*elimina_coleccion.py*"*
  En este *script* se escribirá el código para eliminar la colección *libros,* para ello se utilizará el método *drop().*

Quinta parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla Diario de una programadora de los anexos "MPT2A5_DiarioDeUnaProgramadora.pdf"):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Será necesario considerar que el diario correspondiente a esta actividad contenga el nombre completo de la participante y el nombre de la actividad en cuestión.
(Ver apartado de notas).

Notas para apoyar la actividad:

- Se deberá contemplar que la secuencia de pasos para levantar un *cluster* en MongoDB podría cambiar en el futuro, se recomienda corroborar la secuencia de tareas antes de asignarla a las participantes, en dado caso de que varíe, realizar los cambios correspondientes.
- En caso de que a la participante se le dificulte realizar las tareas, deberá leer la documentación de MongoDB, MongoDB

Atlas, o de PyMongo.

Notas (parte 4):

- A partir de este *script,* queda implícito que en los siguiente también se deberá importar *pymongo* y seguido se creará una instancia de *MongoClient(),* la cual se almacenará en la variable *cliente,* al inicio de cada *script.*
- A partir de este *script,* queda implícito que en los siguientes pasos aparte de importar *pymongo* y la creación de la instancia de *MongoClient()* en la variable *cliente* para hacer la conexión al *cluster*, también se deberá incluir la obtención de la base de datos *biblioteca_bd* y de la colección *libros* al inicio de cada *script.*

Notas (Parte 5)**:**

- Es importante no olvidar que este diario se le solicitará a la participante en cada actividad para ser revisado y tomado en cuenta para las posibles modificaciones del desarrollo de las actividades.

Links para aprender más:

- Mongo DB. (Sin fecha). *MongoDB Atlas.* https://www.mongodb.com/cloud/atlas (Agosto, 2020).
- Mongo DB. Documentación.(Sin fecha) *Mongo DB Atlas.* https://docs.atlas.mongodb.com/ (Agosto, 2020).
- PyMongo 3.11.1 Documentation. (Sin fecha). *Tutorial.* https://pymongo.readthedocs.io/en/stable/tutorial.html (Agosto, 2020).
    - *collection- Collection level operations. https://www.google.com/url?q=https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html&sa=D&source=editors&ust=1615422513071000&usg=AOvVaw09PfaAcvf_ClIHYZ4CG0xD* (Agosto, 2021)

Actividad 10: Pongamos MongoDB al módulo “libros”.

<table>
<tr><td colspan="2">Aprendizaje esperado: Agregar los métodos CRUD para el módulo de libros usando MongoDB.</td><td>Duración de la actividad: 3 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Proyecto Atenea.</td><td>Producto:
1. Archivo app.py modificado.</td><td>Evaluación:
1. Corroborar que el CRUD del módulo libros funcione con la conexión a la base de datos sin errores efectuando las pruebas solicitadas.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Integrando CRUD.
Sugerencia de tiempo invertido: 3 horas.

La participante realizará la búsqueda en internet del objeto ObjectId BSON, identificando la función de este y de la clave “_id”, de la cual hace uso el objeto.

La participante realizará las modificaciones al CRUD del módulo libros de su proyecto Atenea para conectarlo a la base de datos biblioteca_bd alojada en la nube, para ello hará lo siguiente:
1. En el archivo app.py agregará dos import, uno para importar pymongo, y otro para importar el objeto objectId de la biblioteca bson.objectid.
2. Realizará la conexión a la base de datos así como lo hizo en el script conectar_bd.py de la actividad 9, utilizando MongoClient(), luego, obtendrá la base de datos biblioteca_bd y la colección libros, utilizando el acceso al estilo de diccionario.</td></tr>
</table>

(Ver apartado de notas).

3. Borrará la lista de diccionarios *libros,* ya que ahora se tiene la base de datos en la nube donde se almacenarán los registros de los libros.
4. En el método *libros_lista(),* utilizará el método *find()* antes del *return* para leer todos los registros de la colección *libros.*
   En la plantilla *libros_lista.html* que se retorna en este método deberá modificar los enlaces de *Detalles* y *Editar* para que ahora reciban el campo *_id* del objeto *ObjectId,* de *libro* es decir: libro['_id'].
5. En el método *libro(),* cambiará la ruta por "/libros/libro/<id>" y dentro del método declarará una consulta en una variable de nombre "mi_query", que funcionará como llave para buscar el registro mediante su id, que sea igual al que se está recibiendo como parámetro en el método, es decir: mi_query = { "_id": ObjectId(id) }
   Luego, utilizará el método *find_one()* pasándole como parámetro la variable que almacena la consulta.
6. En el método *agregar_libro(),* cambiará la sentencia donde se agrega un nuevo registro al diccionario, utilizando ahora el método *insert_one()* en la colección, pasándole como parámetro el diccionario *libro_nuevo.*
7. En el método *editar_libro(),* cambiará la ruta por "/libros/editar/<id>" y dentro del método declarará una consulta en una variable de nombre "mi_query", igual que la consulta del punto número cinco, luego utilizará el método *find_one(),* pasándole como parámetro la variable que almacena la consulta, para encontrar el registro del libro que se va a editar por lo que esto se escribirá antes del bloque *if..else,* dentro del condicional modificara el diccionario *libro_nuevo* para que se utilice el operador $*set* así como se hizo en el *script edita_registro.py* de la actividad 9. Debajo del *if..else* utilizará el método *update_one()* pasándole como primer parámetro la consulta *mi_query* y como segundo parámetro el diccionario *libro_nuevo.*
8. En el método *eliminar_libro(),* cambiará la ruta por "/libros/eliminar/<id>" y dentro del método declarará una consulta "mi_query" igual a la del punto número cinco, luego utilizará el método *delete_one(),* pasándole como parámetro la consulta para eliminar el registro de la colección *libros.*
   (Ver apartado de notas)

Una vez realizado lo anterior:

- Se le solicitará a la participante que realice las pruebas para corroborar que el CRUD del módulo *libros* funcione con la conexión a la base de datos, por lo que hará lo siguiente:
  - Agregará cinco libros.
  - Listará los cinco libros agregados.
  - Mostrará los detalles de algún libro en particular.

        - ❖ Editará la información de al menos un libro.
        - ❖ Eliminará un libro.
    - Reiniciará el servidor donde se muestra el proyecto y responderá la siguiente pregunta: ¿Por qué los cambios en los registros se mantienen aún después de reiniciar el proyecto? Comentará la respuesta con la tallerista o con otras participantes.
    - La participante hará *commit* sobre su repositorio *Atenea* de Github.

Segunda parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla Diario de una programadora de los anexos *"MPT2A5_DiarioDeUnaProgramadora.pdf"*):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Será necesario considerar que el diario correspondiente a esta actividad contenga el nombre completo de la participante y el nombre de la actividad en cuestión.

Nota:

- Es importante no olvidar que este diario se le solicitará a la participante en cada actividad para ser revisado y tomado en cuenta para las posibles modificaciones del desarrollo de las actividades.

Notas (Parte 1):

- Los dos pasos anteriores se escribirán antes de instanciar el objeto *Flask.*

- En la actividad 9 se eliminó la colección *libros* por lo que se deberá agregar nuevamente esta colección en el dashboard de MongoDB Atlas.

Links para aprender más:

- Mongo DB. (Sin fecha) *Mongo DB Atlas* https://www.mongodb.com/cloud/atlas (Agosto, 2020)
- Mongo DB| *Documentación.* (Sin fecha) *Mongo DB Atlas* https://docs.atlas.mongodb.com/ (Agosto, 2020).
    - *BSON Types* https://docs.mongodb.com/manual/reference/bson-types/ (Agosto, 2020).
- PyMongo 3.11.1 Documentation. (Sin fecha) *Tutorial* https://pymongo.readthedocs.io/en/stable/tutorial.html (Agosto, 2020).
- Documentación de PyMongo (Sin fecha) *Collection. Operación de nivel de colección* *https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html&sa=D&source=editors&ust=1615422513076000&usg=AOvVaw3VbxjOww91NlpEF87zAKHl*
- w3big.com.(Sin fecha). *Mongo DB Tutorial avanzado. MongoDB ObjectId* http://www.w3big.com/es/mongodb/mongodb-objectid.html (Agosto,2020).

Actividad 11: Ahora lo mismo pero para el módulo “usuarias”

<table>
<tr><td colspan="3">Filosofía E-PILA R ES: Repasar.
Está comprobado que para adquirir un conocimiento, es indispensable practicarlo un sinnúmero de veces. El repasar una y otra vez lo que se ha aprendido en este módulo, reafirmará los conocimientos, permitiendo que estos permanezcan con mayor facilidad.
Recuerda: la práctica hace a la maestra, por lo que repasar los conocimientos demanda constancia, disciplina y compromiso.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Agregar los métodos CRUD para el módulo de usuarias usando MongoDB.</td><td>Duración de la actividad: 6 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Python v.3.x.x
● Flask v.1.1
● Pymongo
● MongoDB Atlas
● Anexo MPT2A11.pdf
● Archivo base.html del proyecto Atenea.
● Archivo app.py del proyecto Atenea.</td><td>Evidencia:
1. Agregar_usuaria.html
2. Usuarias_lista.html
3. usuarias.html</td><td>Retroalimentación:
● Para el archivo agregar_usuaria.html deberá: -Hacerse uso del archivo base.html para integrar los elementos de la cabecera y estilo.
● El formulario debe desarrollarse con plantillas para formularios incluidas en la documentación de Bootstrap. Puede modificarse el estilo pero la base debe ser tomada de dicho framework, incluso puede ser parecido al de la actividad 6 de este taller.
● Para el archivo usuarias_lista.html deberá:
- Hacer uso del archivo base.html para integrar los elementos de la cabecera y estilo.
- Hacer uso de un bloque for de Jinja.
● Para el archivo usuarias.html deberá:
- Hacer uso de un bloque for en Jinja.
- Omitir mostrar el password de las usuarias.</td></tr>
</table>

		● Para el archivo app.py deberá: - Contener la ruta “/usuarias/agregar” con el método GET y POST. - Contener el método agregar_usuaria() - Hacer uso de la colección usuarias dentro del método agregar_usuaria(). - Hacer uso de la función flash para mostrar alertas. - Verificar que exista un request para el nombre de la usuaria desde el archivo agregar_usuaria.html. - Agregar el registro a la colección usuarias haciendo uso de la función insert_one(). - Hacer uso de la función *redirect()* y *url()* para mostrar las vistas. - Contener la ruta “/usuarias/filtro” - Contener el método usuarias_filtro() - Hacer uso de la función *find()* dentro del método *usuarias_filtro()* - Contener la ruta *“*usuaria/actualiza” - Contener el método *usuaria_actualiza*(). - Hacer uso de la función *update_one()* dentro del método *usuaria_actualiza*(). - Contener la ruta “/usuaria/elimina”. - Contener el método *usuaria_elimina*(). - Hacer uso de la función *delete_one()*.

Desarrollo de la actividad:

Primera parte: Creando registros de mujeres en la historia de mi aplicación.
Sugerencia de tiempo invertido: 5 horas y 30 minutos.
En las actividades anteriores, la participante desarrolló el CRUD correspondiente al módulo *libros* de su proyecto. A continuación, creará ahora el CRUD completo y las vistas correspondientes para el módulo *usuarias*. Para ello deberá desarrollar los siguientes scripts con las especificaciones necesarias para cada uno:

- ***Plantilla* "agregar_usuaria.html"**
    1. Dentro de la carpeta *templates* creará un documento HTML llamado "agregar_usuaria.html", y dentro de este, extenderá la página *base.html*.
    2. En este mismo archivo (*agregar_usuaria.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
        - Creará un formulario para la petición de los datos necesarios para registrar a una usuaria: nombre, apellido paterno, apellido materno, correo electrónico y número hash de la contraseña. Puede tomarse como guía el formulario creado en la actividad 6 parte 2 de este taller.
        - Utilizará el método *POST* para el formulario.
        - Creará un botón que permita el envío de datos a la aplicación *app.py*
    3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "agregar_usuaria()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *agregar_usuaria.html* y la variable que contiene el nombre de la biblioteca. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
        - Escribirá la ruta "/usuarias/agregar" con la opción de métodos *GET* y *POST*.
        - Accederá a la colección *usuarias.*
        - Verificará que el método con el que se ha accedido a la vista es el método POST.
        - De cumplirse la condición anterior, verificará que exista un *request* para el dato correspondiente a nombre.
        - De no ser así, mostrará un mensaje con la función *flash()* que mencione que el nombre es un dato necesario, ya que sin este no tendría sentido el almacenamiento de una usuaria.
        - De existir el *request* para la usuaria, creará un diccionario en el que se almacenen los datos obtenidos del formulario, a través de la función *request()* y el campo que se desea guardar en cada una de las llaves del diccionario, por ejemplo: "nombre" : *request.form*('nombre')

- Una vez que se hayan agregado al diccionario todos los elementos del formulario, utilizará la función *insert_one()* para agregar el diccionario a la colección *usuarias* de la base de datos.
- Con ayuda de la función *redirect()* y *url_for()*, redireccionará a la/el usuario a la misma vista (*agregar_usuaria.html*) para que continúe registrando usuarias.

- ***Plantilla* "usuarias_lista.html"**
  1. Dentro de la carpeta *templates* creará un documento HTML llamado "usuarias_lista.html", y dentro de este, extenderá la página *base.html*.
  2. En este mismo archivo (*usuarias_lista.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
     - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*
     - Con la sintaxis de un bloque *for* en *Jinja,* recorrerá los diccionarios que se encuentran en la lista *users* que será pasada como argumento en *app.py*.
     - Por medio de etiquetas HTML, mostrará los datos de cada usuaria, excepto el *password* de registro. Para lo anterior, recorrerá los elementos de cada diccionario obtenido con el bloque *for*, haciendo uso de la clave con la que definió los contenidos del diccionario.
  3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "usuarias_lista()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *usuarias_lista.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene todas las usuarias de la base de datos. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
     - Escribirá la ruta "/usuarias/lista".
     - Accederá a la colección *usuarias.*
     - Creará una lista en donde almacene los datos de todas las usuarias registradas en la base de datos. Lo anterior lo realizará con ayuda de la función *find()* de *Pymongo*.

- ***Plantilla* "usuarias.html"**
  1. Dentro de la carpeta *templates* creará un documento HTML llamado "usuarias.html", y dentro de este, extenderá la página *base.html*.
  2. En este mismo archivo (*usuarias.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
     - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*

- Con la sintaxis de un bloque *for* en *Jinja,* recorrerá los diccionarios que se encuentran en la lista *ussers* que será pasada como argumento en *app.py*.
- Por medio de etiquetas HTML, mostrará los datos de cada usuaria, excepto el password. Para lo anterior, recorrerá los elementos de cada diccionario obtenido con el bloque *for* de *Jinja*, haciendo uso de la clave con la que definió los contenidos del diccionario.

3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "usuarias_filtro()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *usuarias.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene todas las usuarias de la base de datos. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
    - Escribirá la ruta "usuarias/filtro".
    - Accederá a la colección *usuarias.*
    - Creará una lista en donde almacene los datos de todas las usuarias registradas en la base de datos (lista *ussers*) que cumplan con cierto criterio de búsqueda, por ejemplo, todas las usuarias que estén registradas con el nombre de "Laura". Lo anterior lo realizará con ayuda de la función *find()* de *Pymongo* a la cual se le pasará como argumento el criterio de búsqueda.
      (Ver apartado de notas).

Una vez que se hayan creado los scripts anteriores, la participante:

1. Para actualizar el registro de una usuaria, escribirá en el archivo *app.py* el método "usuaria_actualiza()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *usuaria.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene todas las usuarias de la base de datos que se desean actualizar (lista *ussers*). En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
    - Escribirá la ruta "usuaria/actualiza".
    - Accederá a la colección *usuarias.*
    - Declarará una variable que contenga la tupla que funcionará como llave para buscar el registro que se desea actualizar, por ejemplo: mi_query = {"nombre" : "Laura"}
    - Guardará en una lista los datos de todas las usuarias registradas en la base de datos (lista *ussers*) que cumplan el criterio de búsqueda de la variable definida en el punto anterior.
    - Guardará en una lista los datos nuevos para el registro a actualizar.
    - Utilizará la función *update_one()* para actualizar el registro haciendo uso de las variables declaradas en los puntos previos a este.

(Ver apartado de notas).

2. Para eliminar el registro de una usuaria, escribirá en el archivo *app.py* el método “usuaria_elimina()”, que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *usuarias_lista.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene todas las usuarias de la base de datos (*ussers*). En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
    - Escribirá la ruta “/usuaria/elimina”
    - Accederá a la colección *usuarias.*
    - Declarará una variable que contenga la tupla que funcionará como llave para buscar el registro que se desea eliminar, por ejemplo: mi_query = {“nombre” : ”Atenea”}
    - Eliminará el registro con ayuda de la función *delete_one()*, haciendo uso de la tupla del punto anterior.
    - En una lista llamada *ussers*, guardará todas las usuarias registradas en la base de datos.
      (Ver apartado de notas).
3. Por último, la participante creará un *script* que permita eliminar la colección *usuarias* de la base de datos, para esto deberá hacer uso de la función *drop()* sobre la colección.

En caso de que no exista ningún error de sintaxis o lógica en el desarrollo de esta parte de la actividad, procederá a hacer *commit* sobre su repositorio *Atenea* de *GitHub* para mantener los cambios realizados en el proyecto.

Segunda parte: Probando mi trabajo.
Sugerencia de tiempo invertido: 30 minutos.

La participante probará el CRUD realizado para el módulo de usuarias de la parte uno de esta actividad, para ello:

1. Agregará al menos tres usuarias con sus datos completos para el registro.
2. Visualizará que se hayan agregado las tres usuarias a la lista de usuarias.
3. Modificara el nombre de la primera usuaria que agrego.
4. Eliminará a la tercera usuaria que agrego.
5. Volverá a visualizar los cambios en la lista de usuarias.

Verificará que todos los pasos se hayan realizado correctamente, de no ser así modificará la parte del código que esté generando la falla.

Tercera parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.

La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla *Diario de una programadora* de los anexos MPT2A5_DiarioDeUnaProgramadora.pdf):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Para este momento, se hará en conjunto con la participante un diálogo de retroalimentación, tomando como base todos los diarios elaborados desde la actividad 5 hasta la presente, de tal manera que se discutan las posibles dudas técnicas que aún persistan sobre dichas actividades, así como posibles sugerencias para modificar la dinámica y los ejercicios de las actividades.

Notas para apoyar la actividad:

- Parte 1:
    - Los nombre de usuarias que se registren no deben contener datos personales reales por seguridad.
    - Se sugiere que se utilicen nombres de mujeres escritoras para simular las usuarias dentro de la aplicación.

Nota (Parte 1)**:**

- Este método (usuarias_filtro) no contiene una vista en el front-end de la aplicación, por lo tanto se sugiere que la participante la desarrolle bajo sus criterio de diseño y haciendo uso de lo aprendido durante este taller.
- Este método (usuaria_actualiza) no contiene una vista en el front-end de la aplicación, por lo tanto se sugiere que la participante la desarrolle bajo sus criterio de diseño y haciendo uso de lo aprendido durante este taller.
- Este método (usuaria_elimina)no contiene una vista en el front-end de la aplicación, por lo tanto se sugiere que la participante la desarrolle bajo sus criterio de diseño y haciendo uso de lo aprendido durante este taller.

Actividad 12: Terminamos con el módulo “préstamos”.

<table>
<tr><td colspan="3">Filosofía E-PILA R ES: Repasar.
Está comprobado que para adquirir un conocimiento, es indispensable practicarlo un sinnúmero de veces. El repasar una y otra vez lo que se ha aprendido en este módulo, reafirmará los conocimientos, permitiendo que estos permanezcan con mayor facilidad. Recuerda: la práctica hace a la maestra, por lo que repasar los conocimientos demanda constancia, disciplina y compromiso.</td></tr>
<tr><td colspan="2">Aprendizaje esperado: Agregar los métodos CRUD para el módulo de préstamos usando MongoDB.</td><td>Duración de la actividad: 8 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td></td><td>Evidencia:
Fichas de consultas sobre join.
Plantillas “agregar_prestamo.html”, “prestamos_lista.html”, “prestamo_ eliminar.html” y “prestamo_actualizar.html”</td><td>Retroalimentación:
Revisar que la información sobre el concepto de join sea correcta y fundamentada en cuentas.
Verificar el código de CRUD que realice las acciones de agregar, listar, borrar y actualizar.</td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Unirse siempre es bueno.
Sugerencia de tiempo invertido: 1 hora.

La participante realizará la búsqueda en internet del concepto join en bases de datos. Leerá las páginas necesarias e incluso podrá apoyarse de material visual como videos que le permitan entender la funcionalidad de esta sentencia.
La información recabada deberá ser anotada en fichas de consulta o en el organizador de su preferencia.
Por último, investigará la sintaxis para realizar un join en MongoDB Atlas para ponerlo en práctica en su aplicación.

Segunda parte: Pidamos prestado.
Sugerencia de tiempo invertido: 6 horas y 30 minutos.</td></tr>
</table>

La participante creará ahora el CRUD completo y las vistas correspondientes para el módulo *préstamos*. Para ello deberá desarrollar los siguientes scripts con las especificaciones necesarias para cada uno:

- ***Plantilla* "agregar_prestamo.html"**
    1. Dentro de la carpeta *templates* creará un documento HTML llamado "agregar_prestamo.html", y dentro de este, extenderá la página *base.html*.
    2. En este mismo archivo (*agregar_prestamo.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
        - Creará un formulario para la petición de los datos necesarios para registrar un préstamo: nombre de la usuaria, título del libro y fecha actual. El nombre y el título deberán ser elegidos por medio de un combo que será desplegado con la iteración de una lista para cada uno (libros y usuarias) que serán obtenidas de *app.py*. La fecha deberá obtenerse a través de la etiqueta input<type="date"> de HTML. Puede tomarse como guía el formulario creado en la actividad 6 parte 2 de este taller.
        - Utilizará el método *POST* para el formulario.
        - Creará un botón que permita el envío de datos a la aplicación *app.py*
    3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "agregar_prestamo()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *agregar_prestamo.html,* una lista que contenga todas las usuarias registradas en el sistema (lista *users*), una lista que contiene los libros registrados en el sistema (lista *books*) y la variable que contiene el nombre de la biblioteca. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
        - Escribirá la ruta "/prestamo/agregar" con la opción de métodos *GET* y *POST*.
        - Accederá a las colecciones *usuarias, libros* y *préstamos.* Deberá utilizarse una variable por cada colección.
        - Creará tres listas: una para almacenar los registros existentes en la colección *usuarias (*lista *users)*, otra para los registros existentes en la colección *libros (*lista *books)* y una más para la colección de préstamos (lista *préstamo*). Esto lo hará con ayuda de la función *find()* de *pymongo*.
        - Verificará que el método con el que se ha accedido a la vista es el método POST.
        - De cumplirse la condición anterior, verificará que exista un *request* para el dato correspondiente a usuaria.
        - De no ser así, mostrará un mensaje con la función *flash()* que mencione que el nombre de la usuaria es un dato necesario, ya que sin este no puede registrarse un préstamo.
        - De existir el *request*, con los datos obtenidos del formulario deberá obtener el id del libro que corresponde al título ingresado y el id de la usuaria que se eligió. Esto deberá hacerse por medio de un *query* para la función *find()* de la base de datos.

- Creará un diccionario en el que se almacenen los siguientes datos: id de la usuaria, id del libro, fecha de préstamo y fecha de devolución, por ejemplo: “fecha” : *request.form*(‘fecha’)
- Para la fecha de entrega del libro, se le sumarán 7 días a la fecha de préstamo. Esto puede realizarse con ayuda de la biblioteca *DateTime* de *Python*, la cual permite el manejo de fechas.
- Una vez que se hayan agregado al diccionario todos los elementos del formulario, utilizará la función *insert_one()* para agregar el diccionario a la colección *usuarias* de la base de datos.
- Con ayuda de la función *redirect()* y *url_for()*, redireccionará a la/el usuario a la vista (*prestamos_lista.html*) para que visualice los préstamos existentes en la base de datos.

- ***Plantilla* “prestamos_lista.html”**
  1. Dentro de la carpeta *templates* creará un documento HTML llamado “prestamos_lista.html”, y dentro de este, extenderá la página *base.html*.
  2. En este mismo archivo (*prestamos_lista.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
     - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*
     - Con la sintaxis de un bloque *for* en *Jinja,* recorrerá la lista de datos en común entre las colecciones (lista *préstamos común*) que será pasada como argumento en *app.py*.
     - Por medio de etiquetas HTML, mostrará únicamente el nombre de la usuaria que solicitó el préstamo, el nombre del libro y la fecha de devolución. Para lo anterior deberá hacer uso de un *join* a las colecciones usuarias y libros en donde la llave a buscar serán los ids de cada colección y con ello se obtendrán los datos necesarios para la vista.
  3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método “prestamos_lista()”, que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *prestamos_lista.html*, la variable que contiene el nombre de la biblioteca, la lista que contiene todos los libros y la lista que contiene el registro de usuarias. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
     - Escribirá la ruta “/préstamos/lista”.
     - Accederá a la colección *usuarias, préstamos y libros.*
     - Realizará un join para obtener los registros en común entre las tres colecciones, en donde los id's de las colecciones libros y usuarias serán los que fungirán como llave local y llave foránea.
     - Guardará lo obtenido en el *join* dentro de una lista (lista *préstamos_común*).

- ***Plantilla* “préstamo_eliminar.html”**
    1. Dentro de la carpeta *templates* creará un documento HTML llamado “préstamo_eliminar.html”, y dentro de este, extenderá la página *base.html*.
    2. En este mismo archivo (*préstamo_eliminar.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
        - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*
        - Creará una vista parecida a la desarrollada en el documento *prestamos_lista.py* agregandole un botón de eliminar a cada registro mostrado.
        - El botón de eliminar deberá regresar el id del registro en la colección préstamos que se desea eliminar.
    3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método “préstamo_eliminar()”, que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *prestamos_eliminar.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene los registros de los préstamos. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
        - Escribirá la ruta “/préstamo/elimina”.
        - Accederá a la colección *usuarias, préstamos y libros.*
        - Realizará un *join* para obtener los registros en común entre las tres colecciones, en donde los id's de las colecciones libros y usuarias serán los que fungirán como llave local y llave foránea.
        - Guardará lo obtenido en el *join* dentro de una lista (lista *préstamos_común*).
        - Guardará en una variable el id del préstamo que se desea eliminar.
        - Eliminará el registro de la base de datos con ayuda de la función *delete_one()* de *Pymongo.*
        - Usando función *redirect()* y *url_for()*, redireccionará a la/el usuario a la vista (*prestamos_lista.html*) para que visualice los préstamos existentes en la base de datos y verificar que el registro se eliminó de la base de datos.

- ***Plantilla* “préstamo_actualizar.html”**
    1. Dentro de la carpeta *templates* creará un documento HTML llamado “préstamo_actualizar.html”, y dentro de este, extenderá la página *base.html*.
    2. En este mismo archivo (*préstamo_actualizar.html*) utilizará las etiquetas *block* y *endblock* para delimitar las siguientes acciones:
        - Mostrará un mensaje con el nombre de la biblioteca, el cual será obtenido del archivo *app.py*

- Creará una vista parecida a la desarrollada en el documento *prestamos_lista.py* agregándole un botón de actualizar a cada registro mostrado.
- El botón de actualizar deberá regresar el id del registro en la colección préstamos que se desea renovar.

3. Habiendo realizado las acciones anteriores, la participante escribirá en el archivo *app.py* el método "préstamo_actualizar()", que retornará una llamada a la función *render_template()*, pasando como parámetros el documento *prestamos_actualizar.html*, la variable que contiene el nombre de la biblioteca y la lista que contiene los registros de los préstamos. En este mismo método, la participante desarrollará el código correspondiente a las siguientes acciones:
    - Escribirá la ruta "/préstamo/actualiza".-
    - Accederá a la colección de préstamos.
    - Buscará el registro correspondiente al id devuelto por la vista *préstamo_actualiza*().
    - Ingresará a los atributos del registro y actualizará la fecha de entrega, sumandole 7 días más a dicha fecha.
    - Guardará el registro con la función update_one().
    - Usando función *redirect()* y *url_for()*, redireccionará a la/el usuario a la vista (prestamos_lista.html) para que visualice los préstamos existentes en la base de datos y verificar que el registro se actualizó.

Una vez terminados los pasos anteriores, se le solicitará a la participante realizar el registro de tres préstamos con tres libros y tres usuarias distintas, de tal manera que se pueda verificar su funcionalidad. En caso de que no exista ningún error de ejecución en el desarrollo de esta parte de la actividad, procederá a hacer commit sobre su repositorio *Atenea* de *Github* para mantener los cambios realizados en el proyecto.

Tercera parte: Probando mi trabajo.
Sugerencia de tiempo invertido: 30 minutos.
La participante probará el CRUD realizado para el módulo de préstamos de la parte dos de esta actividad, para ello:

1. Agregará al menos dos préstamos de un libro para una usuaria.
2. Visualizará que se hayan agregado los préstamos a la lista de préstamos.
3. Modificará uno de los dos préstamos.
4. Eliminará el préstamo que no modificó.
5. Volverá a visualizar la lista de préstamos.

Verificará que todos los pasos se hayan realizado correctamente, de no ser así modificará la parte del código que esté generando la falla.

Cuarta parte: Diario de una programadora.
Sugerencia de tiempo invertido: 30 minutos.
La participante continuará creando su diario personal de programación, por medio de responder las siguientes preguntas (se recomienda utilizar la plantilla *Diario de una programadora* de los anexos “MPT2A5_DiarioDeUnaProgramadora.pdf”):

- ¿Qué aprendí?
- ¿Qué me gustó más?
- ¿Cómo me sentí mientras lo aprendí?
- ¿Se me dificultó algo? ¿Cómo lo solucioné?
- ¿Me hubiera gustado cambiar algo de la actividad?

Será necesario considerar que el diario correspondiente a esta actividad contenga el nombre completo de la participante y el nombre de la actividad en cuestión.
(Ver apartado de notas).

Notas para apoyar la actividad (Parte 4):

- Es importante no olvidar que este diario se le solicitará a la participante en cada actividad para ser revisado y tomado en cuenta para las posibles modificaciones del desarrollo de las actividades.

Actividad 13: Si no le pones candado a tu aplicación se roban tus memes.

Filosofía E-PILA **R** ES: **Repasar**
Está comprobado que para adquirir un conocimiento, es indispensable practicarlo un sinnúmero de veces. El repasar una y otra vez lo que se ha aprendido en este módulo, reafirmará los conocimientos, permitiendo que estos permanezcan con mayor facilidad. Recuerda: la práctica hace a la maestra, por lo que repasar los conocimientos demanda constancia, disciplina y compromiso.

Aprendizaje esperado: Agregar los métodos CRUD para el módulo de préstamos usando MongoDB.		**Duración de la actividad:** 7 horas.
Recursos	**Evidencia/producto**	**Retroalimentación/Evaluación**
● Flask-login ● Werkzeug.security ● Archivo *models.py*, *forms.py* y *app.py.* ● Módulo "*models*"	**Evidencia**: ● Código para la colección "Usuarias".	**Retroalimentación:** ● Verificar que la lógica sea correcta. ● Verificar que las vistas que se hayan decidido estén protegidas y no sean accesibles fuera del sistema. ● Verificar que la webapp se ejecute en url 127.0.0.1:5000

Desarrollo de la actividad:

Primera parte: ¿Dónde guardamos los datos de las bibliotecarias?
Sugerencia de tiempo invertido: 6 horas.

Nota: La colección "Usuarias" almacena a las personas que acuden a la biblioteca a pedir un libro, sin embargo, las bibliotecarias serán las encargadas de administrar el sistema y su información estará almacenada en la colección "Users".

1. Crear la colección "Users" con los campos id, name, email, password y is_admin en MongoDB.
2. Crear la clase "Users" con las funciones set_password() y check_password() como se muestra en el archivo "*models.py"* del directorio ejemplo1 dentro del repositorio .
3. Importar "users" del módulo *"models"* desarrollado en el punto anterior.
4. Ya se tiene la lógica que establece una contraseña y su verificación, ahora falta generar un procedimiento para que la bibliotecaria pueda identificarse. Para ello crear una clase LoginForm como se muestra en el archivo "*forms.py*"
5. Crear una plantilla login_form.html que incluya las etiquetas form y los campos email y password; así como también el botón

de *login.*

6. Ahora será necesario decirle a Flask que cuando la bibliotecaria ingrese a la url que termina en http://127.0.0.1:5000/login se ejecute la función login. Verifique el archivo *"app.py"*
7. También es necesario poder agregar una nueva bibliotecaria. Agregue la acción signup, tal y como se muestra en el archivo "*app.py"*.
8. Para que la bibliotecaria salga de su sesión será necesario crear la acción logout. Así como se muestra en el archivo app.py
9. Ahora para proteger la vista index solo a bibliotecarias registradas hay que añadir la restricción @login_required después de @app.route('/')

Segunda parte: Protegiendo todo.
Sugerencia de tiempo invertido: 1 horas.

1. Ingrese a la vista index y comente sus observaciones.
2. Ingrese a la vista http://127.0.0.1:5000/libros_lista/ y comente sus observaciones.
3. En grupos de dos participantes discutan qué vistas deben estar restringidas.
4. Agregue la restricción @login_required a cada vista que considere necesaria.

Referencias:

Lozano.J. (Sin fecha). *Tutorial Flask- Lección 4: Login de usuarios en flask.* https://j2logo.com/tutorial-flask-leccion-4-login/ (Noviembre 2020)

Actividad 14: Aplicación web de la biblioteca.

Filosofía E-PILA **R** ES: **Repasar**
Está comprobado que para adquirir un conocimiento, es indispensable practicarlo un sinnúmero de veces. El repasar una y otra vez lo que se ha aprendido en este módulo, reafirmará los conocimientos, permitiendo que estos permanezcan con mayor facilidad. Recuerda: la práctica hace a la maestra, por lo que repasar los conocimientos demanda constancia, disciplina y compromiso.

Aprendizaje esperado: Agregar los métodos CRUD para el módulo de préstamos usando MongoDB.		**Duración de la actividad:** 7 horas.
Recursos	**Evidencia/producto**	**Retroalimentación/Evaluación**
1. Código creado hasta este momento.	**Evidencia**: ● Menú para las acciones. ● Datos agregados y préstamos asignados. ● Demostración de sistema.	**Retroalimentación:** ● Verificar que las acciones sean accesibles desde el menú. ● Verificar que el sistema permita la asignación solicitada.

Desarrollo de la actividad:

Primera parte: Enlaces a todas las acciones.
Sugerencia de tiempo invertido: 2 horas.

1. Con los módulos desarrollados incorporar un menú en la plantilla que lleven a cada una de las acciones.

Segunda parte: Pidamos prestado.
Sugerencia de tiempo invertido: 1 hora.

1. Recopilar los datos bibliográficos de 20 libros.
2. Comprobar el funcionamiento integral de todos los módulos de la siguiente manera:
    a. Agregar al sistema 20 registros de libros.
    b. Agregar al sistema 5 usuarias.
3. Realizar préstamos de la siguiente manera:

Usuaria1	Usuaria2	Usuaria3	Usuaria4	Usuaria5
L1	L2	L11	L15	L7
L20	L19	L5	L8	L3
L12	L4	L13	L14	L9
L6	L17		L16	L18
L10				

4. Quitar el préstamo L2 de la Usuaria2. Anote lo observado.
5. Asignar el préstamo L2 a la Usuaria3. Anote lo observado.
6. Quitar el préstamo L19, L4 y L17. Compruebe en la base de datos que Usuaria2 no tiene ningún préstamo. Anote lo observado.
7. Elimine el L18 y anote lo observado. ¿Cuántos libros tiene Usuaria5?

Tercera parte: Presenta tu sistema.

Sugerencia de tiempo invertido: 2 horas.

1. Cada participante deberá realizar una presentación de su proyecto. Deberá mostrar diapositivas que incluya la siguiente información:
    a. Introducción.
    b. Método de desarrollo.
    c. Diagrama del sistema.
    d. Diapositivas con la apariencia de sus sistema.
    e. Muestra de datos cargados

Cuarta parte: El mejor sistema de préstamo en una biblioteca.

Sugerencia de tiempo invertido: 2 horas.

1. En grupos de dos participantes discutan cómo mejorar el sistema.
2. Discutan y contesten lo siguiente: ¿Cómo hacer una acción que permita saber cuántas veces ha solicitado un libro cualquier usuaria?
3. Discutan y contesten lo siguiente ¿Cómo hacer que el sistema notifique un día antes que prestamos van a caducar?
4. El grupo completo realiza una discusión de las mejoras al sistema.

Taller 3: Introducción a ciencia de datos con Python

En honor a **Grace Murray Hopper (1906-1992)** oficial de la marina estadounidense y científica computacional. Realizó sus estudios en el Colegio Vassar y posteriormente asistió a la Universidad de Yale. En 1943 se incorporó a la Marina donde trabajó como programadora en el MARK 1, primera computadora a gran escala en EE UU.

Inventó el primer compilador durante 1952 que es un programa que traduce las instrucciones con palabras en inglés, al lenguaje máquina de un ordenador. Por otra parte, fue la primera en adscribir el término VIRUS. Fue impulsora de la reducción de las fechas a dos dígitos. Ayudó a desarrollar el lenguaje de programación COBOL situado en los negocios para UNIVAC con la primera computadora.

Módulo: Programación.

Competencia del taller: Procesar, manipular, analizar y visualizar colecciones de datos usando el lenguaje de programación python y el ambiente de desarrollo de Notebooks Jupyter.

Actividad 1: Hago un espacio seguro para Python.

<table>
<tr><td colspan="2">Aprendizaje esperado: Identificar los beneficios de usar ambientes virtuales de python.</td><td>Duración de la actividad: 3 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr>
<td>

Hojas de papel u hojas de reuso, bolígrafo, lápices de color.
Editor de Textos.
Pipenv instalado en la computadora.

</td>
<td>
Evidencia:

Fichas de consulta con las preguntas del primer punto de la primera parte de la actividad y la información recabada acerca de los ambientes virtuales.
Preguntas de reflexión acerca de la experiencia de la participante con el uso de los ambientes virtuales en el taller anterior ubicadas en el punto tres de la primera parte de la actividad.
Recuadro respondido acerca de las características, secuencia de comandos, similitudes y diferencias de virtualenv, pipenv y conda.
Cuento sobre los espacios seguros para mujeres y niñas.

Producto:

Scripts programaAmbiente1.py y programaAmbiente2.py en la carpeta misAmbientesVirtuales.
Archivo de texto requirements_ambiente3.txt

</td>
<td>
Retroalimentación:

Las respuestas a las preguntas del punto uno y tres de la primera parte de la actividad serán según la experiencia de la participante, y se sugerirá que se respondan en fichas de consulta.
Las fichas de consulta serán de formato libre.
Las fichas de consulta acerca de los ambientes virtuales deberán abarcar la definición de ambientes virtuales, problema que solucionan, características y ejemplos. La información deberá ser completa y sintética.
Revisar que el cuadro de características, secuencia de comandos, similitudes y diferencias de virtualenv, pipenv y conda, esté respondido de manera completa y sintética.
El cuento acerca de semejanzas entre espacios virtuales y espacio seguros para mujeres, será de formato libre, de preferencia se comentará con alguna participante o con la tallerista si se prestan las condiciones.

</td>
</tr>
</table>

		Evaluación: ● Corroborar que los scripts *programaAmbiente1.py* y *programaAmbiente2.py* ejecuten sin errores lo que se solicita en la actividad y revisar la respuesta a las pruebas de ejecución de ambientes virtuales ubicados en el apartado de pruebas de funcionamiento de los ambientes virtuales en la tercera parte de esta actividad. ● El archivo de texto *requirements_ambiente3.txt* deberá contener la lista de paquetes que se utilizarán para la correcta ejecución de los *scripts.*

Desarrollo de la actividad:

Primera parte: Mis espacios seguros.
Sugerencia de tiempo invertido: 1 hora.

La participante comprenderá las ventajas del uso de los ambientes virtuales. Para ello la participante:

1. Responderá las siguientes preguntas:
    - Durante el taller anterior, ¿trabajaste directamente sobre el sistema operativo de tu computadora?
    - ¿Instalaste *Flask* en ese taller?
    - ¿Cómo realizaste la instalación?
    - Si tu respuesta a la primera pregunta fue no, ¿cómo se conoce el entorno sobre el cual trabajaste? ¿Instalaste *Flask* sobre dicho entorno?
2. A continuación buscará en internet:
    - ¿Qué son los ambientes virtuales de Python?
    - ¿Qué problema soluciona los ambientes virtuales de Python?

- ¿Cuáles son sus características?
- Señala algunos ejemplos de ambientes virtuales.

La información encontrada deberá conjuntarla en fichas de consultas con el formato que prefiera.

3. Para finalizar, reflexionará acerca del uso de los ambientes virtuales desde su propia experiencia en el taller anterior, por lo que deberá responder:
    - ¿Trabajaste en un ambiente virtual en el taller anterior?
    - Si tu respuesta fue sí, continúa con el resto de las preguntas.
    - ¿Calificarías tu experiencia como buena o mala?
    - ¿Qué ventajas crees que te proporcionó trabajar en un ambiente virtual?
    - ¿Te sentiste segura trabajando en un ambiente virtual?
    - ¿Te sentiste en confianza trabajando en un ambiente virtual?
    - ¿Calificarías a los ambientes virtuales como “espacios seguros”?
    - ¿Qué características tendría para ti un espacio seguro?

(Ver apartado de notas).

Segunda Parte: Mi entorno de desarrollo virtual.
Sugerencia de tiempo invertido: 1 hora.

1. La participante realizará una búsqueda de información sobre cómo crear un ambiente virtual utilizando *virtualenv, pipenv y conda*, además de sus características, secuencia de comandos para la creación de cada uno, similitudes y diferencias. Lo recabado se condensará en la siguiente tabla:

	virtualenv	pipenv	conda
Características			
Secuencia de comandos			
Similitudes			

Diferencias			

Tercera Parte: Más de un espacio seguro en un mismo lugar.
Sugerencia de tiempo invertido: 1 hora.

Se le solicitará a la participante la creación de dos *scripts* en *Python* para demostrar el uso de los ambientes virtuales; con dicha finalidad, realizará una búsqueda que le ayudará a identificar para qué sirven las bibliotecas *faker* y *requests* en *Python* y cómo usarlas. Después, realizará lo siguiente:

1. Creará una carpeta de nombre "misAmbientesVirtuales".
2. Dentro de la carpeta creará un *script* de nombre "programaAmbiente1.py" en el cual escribirá la sentencia *import* para importar *faker*, luego inicializará el generador de falsificaciones y le pasará como argumento la configuración regional de español México.
3. Mediante *prints* imprimirá una serie de datos con esta biblioteca, por ejemplo, nombre, email, estado, etcétera.
4. Creará otro script de nombre "programaAmbiente2.py" en el cual escribirá la sentencia *import* para importar *requests*. Posteriormente, obtendrá la página web "https://pilares.cdmx.gob.mx/" con el método *get* de la biblioteca.
5. Mediante un *print* se imprimirá el contenido de respuesta con *text*, es decir, se visualizará en la consola parte del código HTML de la página web de PILARES.
   (Ver apartado de notas).
6. Una vez escritos los *scripts* se le pedirá la creación de un archivo de texto con nombre "requirements_ambiente3" y en este escribirá la lista de paquetes que se utilizarán para la correcta ejecución de los *scripts*.

Para probar el funcionamiento de los ambientes virtuales, la participante realizará lo siguiente:

1. Creará un ambiente con nombre "ambiente1" y lo activará.
2. En este ambiente instalará *faker* con el comando *pip*.
3. Ejecutará el *script programaAmbiente1.py*
4. Desactivará el *ambiente1* y responderá la siguiente pregunta: ¿El script se ejecutó sin ningún problema? Lo ideal sería que la respuesta a esta pregunta fuera un “sí”, en caso de ser contraria, ¿Por qué se ejecutó con problemas? y ¿Cómo puedo solucionarlo?

5. Creará un ambiente de nombre "ambiente2" y lo activará.
6. Ejecutará el *script programaAmbiente2.py* y responderá la siguiente pregunta: ¿Por qué muestra error en la consola?
7. Una vez identificado que el error se genera debido a que en este ambiente no se ha instalado la biblioteca necesaria, instalará *requests* con el comando *pip*.
8. Ejecutará nuevamente el *script programaAmbiente2.py* y se asegurará que no existan errores en la ejecución.
9. Desactivará el ambiente2.
10. Activará el ambiente1 y ejecutará el *script programaAmbiente2.py,* luego, desactivará el *ambiente1* y activará el *ambiente2*, y en este ejecutará el *script programaAmbiente1.py* y responderá la siguiente pregunta:
    ¿Por qué al intentar ejecutar los scripts en los ambientes contrarios existen errores?
11. Creará un ambiente con nombre "ambiente3" e instalará la paquetería desde la lista *requirements_ambiente3.txt.*
12. Ejecutará ambos scripts en este ambiente y responderá la siguiente pregunta: ¿La ejecución de los scripts fue exitosa? Lo ideal sería que la respuesta fuera un “sí”, en caso contrario, la participante revisará la secuencia de pasos y solucionará los errores que se presenten.
13. Las respuestas las comentará con otras participantes o bien, con la tallerista.

Cuarta parte: Crear nuevos espacios seguros.
Sugerencia de tiempo invertido: 30 minutos.

La participante reflexionará acerca de la creación de ambientes virtuales y su semejanza con la creación de espacios seguros para las mujeres. Para ello se le solicitará redactar en una hoja un pequeño cuento que responda a las siguientes cuestiones:

- ¿Cómo imagino que serían espacios donde las mujeres y niñas puedan sentirse seguras, libres y felices?
- ¿Qué pasos podríamos seguir para crear espacios seguros para mujeres y niñas?

De ser posible, la participante compartirá su cuento con otras participantes. En caso contrario, lo compartirá únicamente con la tallerista.

Notas para apoyar la actividad:

- Para la primera parte de la actividad, se recomienda que la participante responda todas las preguntas de reflexión en fichas.

Notas (parte 1)**:**

- Se recomienda orientar a la participante para que la respuesta a la primera pregunta de esta sección sea afirmativa, en caso de que haya trabajado sobre un ambiente virtual en el taller anterior pero no sea consciente de ello.

Notas (parte 3):

- Para el punto uno y dos la participante deberá apoyarse de la documentación de *faker* para escribir el *script*. Para el punto tres y cuatro deberá apoyarse de la documentación de *requests* para escribir el *script*.

Links para aprender más:

- Faker master (Sin fecha) *Welcome to Faker's documentation*! https://faker.readthedocs.io/en/master/ (Septiembre, 2020).
- Requests(Sin fecha).Quickstart. https://requests.readthedocs.io/es/latest/user/quickstart.html (Septiembre, 2020).
- Warby, W. (2018). *Tutorial de Python virtualenv* https://rukbottoland.com/blog/tutorial-de-python-virtualenv/ (Septiembre, 2020).

Actividad 2: Platico con Python

<table>
<tr><td colspan="2">Aprendizaje esperado: Utilizará sus conocimientos de Python para manipular la interfaz interactiva Jupyter Notebook o JupyterLab.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Jupyter Notebook.
● Jupyter Lab.
● MPT3A2_Anexos.pdf</td><td>Evidencia:
1. Apunte sobre comparación entre Jupyter Notebook y JupyterLab.
2. Jupyter Notebook: DocumentacionMarkdown.
3. Jupyter Notebook: Operaciones básicas.
4. Anexos MPT3A2_Anexos.pdf resueltos.

Producto:
1. Entorno virtual creado.
2. Jupyter Notebook: LineaMujeres.</td><td>Retroalimentación:
● El apunte sobre la comparación entre Jupyter Notebook y JupyterLab, deberá estar conformado mínimo por 300 palabras.
● El notebook DocumentacionMarkdown deberá contener al menos una vez el uso de las etiquetas:
● Para agregar encabezados: h1,h2,h3,h4,h5 y h6.
● Uso de cursivas (texto entre asteriscos).
● Uso de negritas (texto entre dobles asteriscos).
● Enlace a una URL.
● Enlace a una imagen.
● El notebook OperacionesBasicas deberá contener los siguientes elementos:
● Dos listas de elementos de nombre “mujeres” y “hombres”.
● El uso de la función min() de Python.
● El uso de la función max() de Python.</td></tr>
</table>

		● La suma de los elementos de cada lista y posteriormente la división del resultado entre el número de los elementos (función *promedio*). Esta será la lógica para la obtención del promedio. ● Los anexos de esta actividad deberán contar con las plantillas 2 y 3 resueltas. **Evaluación:** El *notebook LineaMujeres* no tiene criterios de evaluación dado que la participante sólo deberá crearlo en esta actividad.

Desarrollo de la actividad:

Primera parte: Toma nota.
Sugerencia de tiempo invertido: 30 minutos.

La participante buscará en internet los siguientes puntos:

- ¿Qué es *Jupyter Notebook*?
- ¿Qué es *JupyterLab*?

Posteriormente se le solicitará realizar lo siguiente:

1. A manera de resumen, la participante escribirá en un cuaderno o en una hoja en blanco una comparación de ambos entornos, en un mínimo de 300 palabras. A su resumen deberá agregarle fecha, título y subtítulo (de ser posible, se recomienda que la participante anexe una imagen de cada uno de los entornos).
2. Una vez realizado su resumen o apunte, se solicitará a la participante explorar la interfaz gráfica de cada entorno, e identificar específicamente: celdas, ejecución de celdas, título del notebook y tipos de formatos.

3. A continuación, la participante complementará su apunte escribiendo (bajo un subtítulo diferente) qué observó en cada interfaz y cuáles son los pasos para abrir cada uno de los dos entornos (*JupyterNotebook* y *JupyterLab)*.

Segunda parte: MARKando la documentación.
Sugerencia de tiempo invertido: 1 hora.

La participante llevará a cabo una búsqueda en internet sobre la definición de *Markdown*, su utilidad y sintaxis. A la par de la búsqueda, realizará el registro de la información obtenida en un *Jupyter notebook* nombrado "DocumentacionMarkdown". Organizará su documentación de la siguiente forma (una celda por punto):

1. Colocará un encabezado h1 para mostrar "Documentación de Markdown" como título.
2. Con un encabezado h2 y el uso de una lista, enumerará los tipos de encabezado y cómo utilizarlos.
3. Con un encabezado h3 colocará como título de la celda "Inserción de imágenes" y documentará la inserción de imágenes con *Markdown* (este texto deberá estar en cursivas).
4. Insertará una imagen relacionada a Python y/o *Markdown*.
5. Agregará en negritas un texto que documente cómo agregar un enlace a una página web.
6. Agregará el enlace de las páginas consultadas para la realización de esta actividad.
7. Agregará al menos dos párrafos (utilizará la sintaxis de *Markdown* para párrafos) sobre lo que espera aprender en este taller y si alguna vez ha tenido acercamiento con la aplicación de las matemáticas en el área de cómputo.
8. Buscará en internet qué es *Data Science* y sus principales aplicaciones. Redactará al menos dos párrafos sobre lo recabado y utilizará la sintaxis aprendida desde el punto uno hasta el siete.

Por último la participante realizará la siguiente reflexión:

- ¿Consideras que *Markdown* es sencillo?
- ¿Cuál sería la principal aplicación que le darías en este taller?

Se entablará una conversación con la participante para responder estas últimas preguntas, y en caso de que aún no comprenda que *Markdown* le será de utilidad para documentar su código de forma más estética y organizada, se le orientará a que llegue a esta conclusión.

Tercera parte: Lo mínimo para aprender lo máximo.
Sugerencia de tiempo invertido: 1 hora.

La participante tendrá un acercamiento a algunas operaciones estadísticas mediante *Jupyter Notebook*. Para ello deberá realizar las siguientes instrucciones:

1. Accederá a la página de *Atlas de género* cuyo link de encuentra en los enlaces anexos al final de este documento.
2. En dicha página, seleccionará la pestaña *Educación*, y posteriormente el apartado *Grado promedio de escolaridad*. En este se mostrará el promedio de años de educación escolar cursados por mujeres y por hombres en cada estado de la república.
3. A continuación, se le solicitará a la participante responder las siguientes preguntas (se recomienda utilizar los anexos *MPT3A2_Anexos.pdf)*:
    - ¿Qué entiendes por promedio?
    - ¿Para qué crees que sirve?
    - ¿Cuál es el mayor número de años de educación escolar que tienen las mujeres?¿En qué estado?
    - ¿A qué crees que se deba que en ese estado las mujeres tienen el promedio más alto de años de educación escolar?
    - ¿Cuál es el menor número de años de educación escolar que tienen las mujeres?¿En qué estado?
    - ¿A qué crees que se deba que en ese estado las mujeres tienen el promedio más bajo de años de educación escolar?
    - ¿Qué entiendes por las funciones mínimo y máximo?
    - ¿Para qué crees que sirven las funciones mínimo y máximo?
4. Una vez resueltas las preguntas anteriores, se solicitará a la participante crear un *Jupyter Notebook* llamado “OperacionesBasicas”.
5. En una celda, creará dos listas: "mujeres" y "hombres".
6. En la lista *mujeres*, almacenará el grado promedio de escolaridad de las mujeres en cada uno de los 32 estados de la república.
7. En la lista *hombres*, almacenará el grado promedio de escolaridad de los hombres en cada uno de los 32 estados de la república.
8. A continuación, utilizará las funciones *min()* y *max()* de *Python* para encontrar los respectivos valores de dichas funciones.
9. Así mismo, la participante obtendrá el grado de escolaridad promedio de todas las mujeres y todos los hombres del país. Para ello, desarrollará una función llamada “promedio” que reciba como parámetro las listas de elementos de hombres y mujeres, y retorne el resultado de las operaciones correspondientes.
10. Estos resultados (mínimo, máximo y promedio) deberá asentarlos en una celda a parte en su *notebook*, con su respectivo título y uso de *Markdown*.
11. Posteriormente, en otra celda documentará la sintaxis y uso de las funciones *min()* y *max()*, de igual manera haciendo uso de *Markdown*.

12. Al final de esta parte deberá asegurarse de que todas las celdas se ejecuten sin errores y muestren lo deseado para cada punto.
13. Una vez realizados los pasos anteriores, la participante deberá responder la siguiente pregunta (se recomienda utilizar la plantilla 3 de los anexos *MPT3A2_Anexos.pdf)*:
    - ¿Cuáles crees que son las ventajas de utilizar *Jupyter Notebook* para realizar estas operaciones estadísticas?

Cuarta parte: Un espacio común y seguro para trabajar.
Sugerencia de tiempo invertido: 30 minutos.

La participante creará un entorno virtual para el desarrollo de las actividades posteriores en este taller, para ello realizará las siguientes instrucciones haciendo uso de los aprendizajes adquiridos en la actividad uno de este taller.

1. Creará una carpeta en una ruta de fácil acceso y permitida (se recomienda hacerlo sobre *Desktop*). La carpeta llevará el nombre de “IntroDataScience_<Nombre de la participante>”.
2. En la carpeta creará un ambiente virtual, haciendo uso del comando *virtualenv.*
3. Con *pip* instalará la paquetería *requirements.txt*
4. Creará un *Jupyter Notebook* con nombre “LineaMujeres”.

Para finalizar, la participante realizará una reflexión acerca de las ventajas de trabajar sobre los ambientes virtuales y su símil, los espacios seguros. Con tales fines, se le solicitará retomar la reflexión que realizó en la primera parte de la actividad 1 de este taller, y a continuación responder las siguientes preguntas:

- ¿Por qué es útil trabajar con ambientes virtuales?
- ¿Por qué es importante tener espacios seguros para mujeres y niñas?

Notas para apoyar la actividad:

- Para la primera parte de la actividad, se recomienda que la participante haga uso de cualquier elemento creativo como colores, imágenes, diferentes tipografías, etcétera, para elaborar su apunte.
- En caso de que a la participante se le dificulte responder las preguntas del punto 3 de la tercera parte, se recomienda que se apoye en una búsqueda en internet para aclarar posibles confusiones.

- Para la tercera parte de la actividad, se recomienda que la participante consulte en la documentación de listas de Python el uso de las funciones mín() y máx().
- Para la tercera parte de la actividad, en caso de que la participante desconozca la forma de obtener un promedio, se le orientará a la búsqueda de información para la obtención del conocimiento necesario para la actividad.

Links para aprender más:

- Brugués, A. (Sin fecha).*Tutorial de Jupyter Notebook*. https://www.programaenpython.com/miscelanea/tutorial-de-jupyter-notebook/#Crear_un_nuevo_Notebook (Septiembre, 2020).
- Cabrera, E. y Díaz Elena. (Sin fecha). *Manual de uso de Jupyter Notebook para aplicaciones docentes*. *Universidad Complutense de Madrid.* https://eprints.ucm.es/48304/1/ManualJupyter.pdf (Septiembre, 2020).
- Cristobal, J. (Sin fecha). *Qué es Markdown*. https://markdown.es/ (Septiembre, 2020).

Actividad 3: ¿Dónde viven los datos?

<table>
<tr><td colspan="2">Aprendizajes esperados :
Listar fuentes de datos, sus formatos y procedimiento para obtenerlos.
Visualizará una demostración de cómo cargarlos y desplegarlos en su notebook.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Hojas de papel, bolígrafo, lapices de color
● Jupyter Notebook
● Jupyter Lab.
● WPS spreadsheets, librecalc o libreoffice</td><td>Evidencia:
1. Jupyter Notebook: CuadernoDeEstadistica con el apartado Presentación, y las preguntas que se solicitan respondidas.
2. Jupyter Notebook: CargandoArchivo.
3. Preguntas respondidas acerca de la información de interés público en la primera parte de la actividad.
4. Preguntas respondidas acerca de la exploración de fuentes de datos públicas, ubicadas en el punto 4 de la primera parte de la actividad.
5. Recuadro de las fuentes de datos públicas, con nombre de la base de datos y tipo de archivo disponible para descargar, ubicado en el punto 5 de la primera parte de la actividad.
6. Recuadro resuelto con las características, similitudes y diferencias de los archivos con extensiones .csv, .json, .excel y .txt.</td><td>Retroalimentación:
● El notebook CuadernoDeEstadistica deberá contener una definición de Ciencia de datos y las respuestas correspondientes a las cinco preguntas que se enuncian en la cuarta parte de la actividad.
● El notebook CargandoArchivo deberá tener cargados los datos del archivo que la participante descargó en el punto 2 de la tercera parte de la actividad. Así mismo, dichos datos deberán poder ser visualizados.
● Las preguntas en el punto dos de la primera parte de la actividad, deberán responderse a partir del conocimiento del Derecho de Acceso a la Información Pública, previamente consultado, en caso de que se dificulte dar respuesta a estas preguntas auxiliar a la participante para llegar a conclusiones dentro el marco del conocimiento de sus derechos.</td></tr>
</table>

		● Las preguntas en el punto cuatro de la primera de la actividad se responderán de acuerdo a la observación de los portales de fuentes de datos públicas sugeridos, corroborar que se use la palabra clave "mujeres". ● El recuadro ubicado en el punto 5 de la primera parte de la actividad, deberá estar respondido de manera completa y la información deberá coincidir con lo que arroja el portal en el momento. ● La tabla ubicada en la segunda parte de la actividad, deberá responderse de manera completa.

Desarrollo de la actividad:

Primera parte: ¿Datos en dónde y para qué?
Sugerencia de tiempo invertido: 1 hora.

La participante tendrá acercamiento a distintas fuentes de datos. Para ello, deberá realizar las siguientes instrucciones:

1. Se le proporcionará a la participante una lista de enlaces a páginas de internet (previa y cuidadosamente seleccionadas), donde se defina qué es el Derecho de Acceso a la Información Pública (se recomienda utilizar los enlaces anexos).
2. La participante deberá explorar dichas páginas, y responder posteriormente:
    - ¿Qué es la “información de interés público”?
    - ¿Cómo me favorece a mí como ciudadana tener acceso a la información de las entidades públicas?
    - ¿Qué tipo de información pública puede estar relacionada con los problemas de las mujeres?
3. Posteriormente la participante explorará las siguientes fuentes de datos públicas:
    - Datos Abiertos de México: https://datos.gob.mx/
    - Portal de datos de la Ciudad de México: https://datos.cdmx.gob.mx/pages/home/

- Plataforma cívica de datos abiertos: http://datamx.io/

4. En cada uno de los anteriores portales, la participante buscará la palabra clave “mujeres”, y contestará lo siguiente:
   - ¿Qué bases de datos arroja la búsqueda?
   - ¿De qué temas tratan dichas bases de datos?
   - ¿Cómo se relacionan tales temas con las mujeres?
   - ¿Cómo crees que se relacionan esos temas con el tipo de fuente (a nivel país, a nivel Ciudad de México, a nivel ciudadanía)?
5. Por último, la participante seleccionará en cada uno de los tres portales, una base de datos de las arrojadas al buscar la palabra “mujeres”, y en un recuadro (como el que se muestra a continuación) escribirá la información que se solicita:

	Nombre de la base de datos	**Tipo de archivo disponible para descargar.**
Datos Abiertos de México		
Portal de datos de la Ciudad de México		
Datos generados por la ciudadanía		

Segunda parte: Datos en diferentes presentaciones.
Sugerencia de tiempo invertido: 30 minutos.

Las participantes realizarán una búsqueda de información acerca de las extensiones de archivos csv, json, excel, y txt, sus características, similitudes y diferencias. Lo recabado se condensará en la siguiente tabla:

	csv	json	excel	txt
Características				

Similitudes				
Diferencias				

Tercera parte: Los mismos datos, vistos de diferente manera.
Sugerencia de tiempo invertido: 1 hora.

1. La participante realizará una búsqueda de información acerca de la biblioteca *csv* de Python para identificar para qué sirve y cómo se utilizaría para cargar y leer un archivo con este tipo de extensión, para ello se podrá apoyar de la documentación disponible en los *links para aprender más* ubicados al final de la actividad.
2. Descargará algún archivo de las fuentes de datos que consultó en la primera parte de la actividad.
3. Creará un *Jupyter Notebook* llamado *"CargandoArchivo"*.
4. En una celda del *notebook* escribirá el código necesario para cargar y leer los datos del archivo descargado utilizando la biblioteca investigada en el punto número uno. Se deberá mostrar en el *notebook* todos los datos del archivo.
5. Abrirá el archivo descargado con algún software disponible en su computadora, por ejemplo, *wps spreadsheets, libreoffice* o *librecalc*, o desde la consola con el comando *less* y realizará una comparativa de lo que observa de las diferentes formas de leer el archivo, para ello responderá en su mismo *notebook* la siguiente pregunta: ¿Qué diferencia hay entre visualizar los datos con su *notebook* a visualizarlos con la consola o con *libreoffice/librecalc*?

Cuarta parte: ¿Qué trabajaré en este taller?
Sugerencia de tiempo invertido: 30 minutos.

La participante creará un nuevo *Jupyter Notebook* con el nombre "CuadernoDeEstadistica". Éste le será de utilidad para recopilar y condensar toda la información que encuentre acerca de operaciones y conceptos de Estadística a lo largo de todo el taller. En dicho *Notebook*, llevará a cabo las siguientes instrucciones:

1. Haciendo uso de *Markdown*, colocará un encabezado h1 con la leyenda "Mi cuaderno de estadística".
2. En una celda aparte, agregará el subtítulo "Presentación", y en él realizará una reflexión acerca de la Ciencia de datos, su utilidad y cómo lo trabajará a lo largo de este taller. En este sentido se le solicitará a la participante retomar la definición de

DataScience que escribió en su *Notebook DocumentacionMarkdown* elaborado en la actividad 2 del presente taller; dicha definición deberá copiarla y pegarla en este apartado "Presentación", y aquí mismo responderá las siguientes preguntas:

- ¿Cuál es la relación entre *Big Data* y *Data Science*?
- ¿Qué es análisis de datos estadísticos?
- ¿Qué es interpretación de datos estadísticos?
- ¿Qué lenguajes de programación se utilizan para la Ciencia de datos?
- ¿Qué espero aprender en este taller?

Notas para apoyar la actividad:

- Para la primera parte de la actividad, se recomienda que la participante responda todas las preguntas de reflexión en fichas.

Links para aprender más:

- IIDH Instituto Interamericano de Derechos Humanos. (Sin fecha). *¡El derecho a la información en acción*! https://www.iidh.ed.cr/derecho-informacion/#:~:text=El%20derecho%20a%20la%20informaci%C3%B3n%20es%20un%20derecho%20humano%2C%20componente,libertad%20de%20pensamiento%20y%20expresi%C3%B3n.&text=El%20derecho%20al%20acceso%20a,de%20cuentas%20de%20las%20autoridades (Septiembre, 2020).
- Info Instituto de Transparencia, Acceso a la Información Pública, Protección de Datos Personales y Rendición de Cuentas de la Ciudad de México. (Sin fecha). *¿Qué es el Derecho de Acceso a la Información Pública?* http://www.infodf.org.mx/index.php/solicita-informacion-publica/%C2%BFqu%C3%A9-es-el-acceso-a-la-informaci%C3%B3n-p%C3%BAblica.html (Septiembre, 2020).
- Python 3.9.1 Documentation (Sin fecha). *csv- CSV File Reading and Writing.* https://docs.python.org/3/library/csv.html (Septiembre, 2020).
- Vaati, Esther. (2017). *Cómo Leer y Escribir Archivos CSV en Python.* https://code.tutsplus.com/es/tutorials/how-to-read-and-write-csv-files-in-python--cms-29907 (Septiembre, 2020).

Actividad 4: Unos pandas para mi Python.

<table>
<tr><td colspan="2">Aprendizaje esperado: Cargar y manipular datos desde la librería especializada en dataframes Pandas.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Notebook LineaMujeres.
● Biblioteca Pandas de Python.
● Python v.3.x.x.</td><td>Evidencia:
1. Organizador gráfico, fichas o resumen con las preguntas del punto 1 de la primera parte.
2. Notebook: CuadernoDeEstadistica con los apartados Conceptos y Ejemplos.
3. Anexos MPT3A4_Anexos.pdf resueltos.

Producto:
1. Notebook: LineaMujeres con dataframe cargado y apartado Ciencia de datos actividad 4.</td><td>Retroalimentación:
● El organizador gráfico, fichas o resumen deberá contar con las respuestas correspondientes a las preguntas del punto 1 de la primera parte de la actividad.
● El apartado Conceptos del notebook CuadernoDeEstadística, deberá contar con la definición de los siguientes conceptos:
● Promedio.
● Mínimo y máximo.
● Media.
● Moda.
● Desviación estándar.
● Percentiles.
● Tipos de datos: numéricos, fechas y categóricos.
● El apartado Ejemplos del mismo notebook deberá contar con un ejemplo de cada uno de los conceptos anteriores.

Evaluación:
La manipulación del dataframe del notebook LineaMujeres deberá realizarse</td></tr>
</table>

		con el siguiente orden de sentencias, y documentarse en el apartado *Ciencia de datos actividad 4*: 1. import pandas as pd 2. <dataframe> = pd.read_csv("linea-mujeres.csv") 3. len(<dataframe>) 4. <dataframe>.columns 5. <dataframe>.dtypes 6. <dataframe>.head(6) 7. <dataframe>.tail(6) 8. <dataframe>.info() 9. <dataframe>.iloc[7] 10. <dataframe>.loc[:,'SERVICIO'] 11. serie = pd.Series(<dataframe>.loc[:,'DÍA_ALTA']) 12. serie.describe() 13. <dataframe>.describe() Dentro del notebook *LineaMujeres* también deberá visualizarse la documentación de cada función utilizada haciendo uso de *Markdown*.

Desarrollo de la actividad:

Primera parte: Pandas estadísticos.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante visualizará y manipulará los datos del *dataframe* que cargará en su *notebook*. Para ello, la participante:

1. Buscará en internet:

- ¿Qué es y para qué sirve la biblioteca *Pandas* de Python?
- ¿Qué es un *dataframe*?
- ¿Qué es una serie de *Python* y cuál es su estructura?

Se recomienda que la participante escriba la información encontrada en algún tipo de organizador gráfico, en fichas o a modo de resumen.

2. Posterior a la búsqueda de información, se le proporcionará a la participante la siguiente lista de funciones, que utilizará para manipular el *dataframe*:
   - *tail*()
   - *iloc*()
   - *pd.Series*()
   - *describe*()
   - *head*()
   - *read_csv*()
   - *len*()
   - *info*()
   - *dtypes*()
   - *loc*()
   - *columns*
3. Una vez que la participante cuente con la lista anterior, se le solicitará llevar a cabo la secuencia de instrucciones que se presenta a continuación. Ella deberá seleccionar la función que crea necesaria para realizar cada uno de los pasos, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se motivará a la participante para buscar en internet las soluciones pertinentes a sus dudas, así como apoyarse en la estrategia de prueba y error hasta que logre completar los pasos uno por celda.
   1. Abrirá el *notebook LineaMujeres* creado en la actividad 2 de este taller.
   2. Descargará el archivo de extensión csv de la página https://datos.cdmx.gob.mx/explore/dataset/linea-mujeres/export/.
   3. Importará la biblioteca *Pandas* (función a utilizar: *import pandas as pd*).
   4. Cargará el archivo csv para obtener un *dataframe* (función a utilizar: *pd.read_csv*).
   5. Mostrará el tamaño del *dataframe* (función a utilizar: *len*).
   6. Mostrará el nombre de las columnas del *dataframe* (función a utilizar: *columns*).
   7. Visualizará los tipos de datos de la información que contiene el *dataframe* (función a utilizar: *dtypes*).

8. Mostrará los índices, el nombre de las columnas, cantidad de nulos y tipos de datos del *dataframe*, con una sola (función a utilizar: *info*).
9. Mostrará las seis primeras filas del *dataframe* (función a utilizar: *head*).
10. Mostrará las seis últimas filas del *dataframe* (función a utilizar: *tail*).
11. Mostrará todos los datos del registro número siete del *dataframe* (función a utilizar: *iloc*).
12. Mostrará todos los datos de la columna SERVICIO (función a utilizar: *loc*).
13. Creará una serie que contenga los datos de la columna DÍA_ALTA (función a utilizar: *pd.Series*).
14. Con una sola función obtendrá información relevante de la serie como el tipo de dato que contiene, el elemento mínimo, elemento máximo y la media (función a utilizar: *describe*).
15. Obtendrá la misma información del punto anterior pero ahora para cada columna del *dataframe*. Es decir, aplicará la función del punto anterior pero al *dataframe* (función a utilizar: *describe*).

4. Una vez realizados todos los puntos, la participante deberá documentar en su *notebook* de *LineaMujeres* la finalidad de cada función, así como sus parámetros correspondientes, por lo que utilizará *Markdown* con el estilo que más prefiera (se recomienda hacer uso de las plantillas 2 y 3 de los anexos *MPT3A4_Anexos.pdf*).
5. Por último, se le invitará a contestar las siguientes preguntas:
    - ¿Ya conocías Línea Mujeres?
    - ¿Alguna vez has llamado? ¿Con qué motivo?
    - De todas las temáticas que hay en el *dataframe*, ¿crees que podrías llamar por alguna de ellas?
    - ¿De qué manera crees que esta herramienta pueda beneficiar a las mujeres en México?

Segunda parte: Mis estadísticas.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Se le solicitará a la participante retomar su *notebook CuadernoDeEstadistica* creado en la actividad anterior, y llevar a cabo las siguientes instrucciones:

1. Agregará un nuevo apartado bajo el subtítulo “Conceptos”. En este anotará las siguientes categorías, de las cuales tendrá que anexar su definición y sus correspondientes operaciones matemáticas:
    - Promedio.
    - Mínimo y máximo.
    - Media.

- Moda.
- Desviación estándar.
- Percentiles.
- Tipos de datos: numéricos, fechas y categóricos.

2. Finalmente para complementar su apunte, en un apartado distinto con nombre “Ejemplos”, agregará un ejemplo del uso en su día a día de cada uno de los conceptos que buscó.

Notas para apoyar la actividad:

- La documentación de los pasos a realizar en la primera parte de la actividad, así como del uso y parámetros de las funciones, deberá llevarse a cabo en su notebook LineaMujeres en un nuevo apartado con el subtítulo “Ciencia de datos actividad 4”.
- Para la primera parte de la actividad, se recomienda cuidar que la participante utilice una celda para cada función en su notebook al momento de documentar la finalidad y parámetros de las funciones.
- Para la primera parte de la actividad, se recomienda consultar la documentación de Pandas.
- Una vez finalizada la primera parte de la actividad, se recomienda motivar a la participante a continuar explorando el dataframe con las funciones aplicadas a otras columnas para obtener más información y practicar.

Links para aprender más:

- Lopez, R. (2016). *Análisis de datos categóricos con Python*. https://relopezbriega.github.io/blog/2016/02/29/analisis-de-datos-categoricos-con-python/ (Septiembre, 2020).
- Moya, R. (2015). *Pandas en Python, con ejemplos Parte I- Introducción.* https://jarroba.com/pandas-python-ejemplos-parte-i-introduccion/ (Septiembre, 2020).
- Pandas. (Sin fecha). *DataFrame.* https://pandas.pydata.org/pandas-docs/stable/reference/frame.html (Septiembre, 2020).
    - *User Guide.* https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html (Septiembre, 2020).

Actividad 5: Lo más común.

<table>
<tr><td colspan="2">Aprendizaje esperado: Identificar los valores más comunes de una columna y su visualización como histograma.</td><td>Duración de la actividad: 4 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Notebook: LineaMujeres.
● Biblioteca Pandas de Python.
● Python v.3.x.x.</td><td>Evidencia:
1. Notebook: CuadernoDeEstadistica con el apartado Gráficas y estadística.
2. Anexos MPT3A5_Anexos.pdf resueltos.

Producto:
1. Notebook: LineaMujeres con los apartados Ciencia de datos actividad 5 e Interpretación de datos A5.</td><td>Retroalimentación:
El apartado Conceptos del notebook CuadernoDeEstadística, deberá contar con la documentación de la siguiente información:
● Uso de gráficas en estadística.
● Tipos de gráficas.
● Uso de la biblioteca Matplotlib de Python.
● Pasos para realizar las gráficas de barras y pastel con Matplotlib, y los parámetros que reciben las funciones a utilizar para ello.

Evaluación:
La manipulación del dataframe del notebook LineaMujeres deberá realizarse con el siguiente orden de sentencias, y documentarse en el apartado Ciencia de datos actividad 5:
● Pregunta 1: value_counts()
● Pregunta 2: value_counts()
● Pregunta 3: value_counts()
● Pregunta 4: value_counts()
● Pregunta 5: value_counts()</td></tr>
</table>

		● Pregunta 6: *value_counts() y str.contains()* ● Pregunta 7: *nlargest* ● Pregunta 8: *nsmallest* Dentro del notebook *LineaMujeres* también deberá existir el apartado *Interpretación de datos A5*, que deberá contener las respuestas a las preguntas de la tercera parte de la actividad.

Desarrollo de la actividad:

Primera parte: ¿En dónde?¿Por qué?
Sugerencia de tiempo invertido: 2 horas.

La participante visualizará y manipulará los datos del dataframe que cargaron en la actividad 4 en su *Notebook LineaMujeres*. Para ello, se le proporcionará a la participante la siguiente lista de funciones, que utilizará para manipular el *dataframe*:

- nlargest()
- value_counts()
- str.contains()
- nsmallest()

Una vez que la participante cuente con la lista anterior, se le solicitará llevar a cabo la secuencia de instrucciones necesaria para poder responder cada una de las preguntas que se presentan. Ella deberá seleccionar la función que crea necesaria para poder obtener la respuesta a las preguntas, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se motivará a la participante para buscar en internet las soluciones pertinentes a sus dudas, así como apoyarse en la estrategia de prueba y error hasta que logre completar los pasos. El código a realizar deberá colocarlo en una celda por pregunta en su *notebook LineaMujeres* dentro de un nuevo apartado con el subtítulo “Ciencia de datos actividad 5”.

1. ¿De qué edad son las mujeres que más reportes hacen en Línea Mujeres? (función a utilizar: *value_counts* sobre Serie)
2. ¿Qué tipo de servicio se solicita más en llamadas telefónicas? (función a utilizar: *value_counts* sobre Serie)
3. ¿En qué mes se reportan más llamadas? (función a utilizar: *value_counts* sobre Serie)
4. ¿Qué ocupación tienen las mujeres que más llaman? (función a utilizar: *value_counts* sobre Serie)

5. ¿Cuántas llamadas que tienen como ORIGEN atención inmediata, están registradas en el dataframe?(función a utilizar: *value_counts* sobre dataframe).
6. ¿Cuántos registros hay de mujeres con Ocupación HOGAR y que hayan registrado VIOLENCIA FAMILIAR como TEMATICA_2? (función a utilizar: *value_counts* y *str.contains*)
7. Anota los datos sobre las 3 personas con mayor edad que han llamado a Línea Mujeres (Servicio, temáticas, origen, etc.). (función a utilizar: *nlargest*)
8. Anota los datos sobre las 3 personas con menor edad que han llamado a Línea Mujeres. ¿Qué notas raro?¿A qué crees que se deba?(función a utilizar: *nsmallest*)

Una vez realizado el código para todos los puntos y habiendo contestado las preguntas, la participante deberá documentar en su *notebook* de *LineaMujeres* la finalidad de cada función que empleó, así como sus parámetros correspondientes (se recomienda hacer uso de los anexos *MPT3A5_Anexos.pdf*).

Segunda parte: Datos y gráficas.
Sugerencia de tiempo invertido: 1 hora 30 minutos.

La participante llevará a cabo cada una de las siguientes instrucciones:

1. Realizará una búsqueda en internet sobre el uso de gráficas en estadística y los tipos de gráficas que existen. Centrará su recopilación de información en las gráficas de pastel y barras, esto lo documentará en su notebook *CuadernoDeEstadistica* creado en la actividad 3 de este taller, en un nuevo apartado de nombre “Gráficas y estadística”.
2. Investigará el uso de la biblioteca *matplotlib* de Python, así como los pasos para realizar las gráficas de barras y pastel con la misma biblioteca, y los parámetros que reciben las funciones a utilizar para ello. Esta información la condensará en el mismo apartado de *Gráficas y estadística* de su *CuadernoDeEstadísitica.*
3. En su notebook *LineaMujeres*, graficará cada uno de los siguientes puntos con base en el código realizado en la parte uno de esta actividad:

    - Gráfica de barras de **ocupaciones** de las mujeres registradas contra la **cantidad de llamadas** recibidas por cada ocupación.
    - Gráfica de barras horizontal de **ocupaciones** de las mujeres registradas contra la **cantidad de llamadas** recibidas por cada ocupación.
    - Gráfica de pastel para la cantidad de registros por **ORIGEN**.

La participante deberá jugar con los parámetros de las gráficas para poder cambiar el color, tamaño, títulos, etiquetas, etcétera.

Tercera parte: ¿Qué me dicen las gráficas?
Sugerencia de tiempo invertido: 1 hora.

Con base en los datos graficados en la segunda parte y las preguntas de la primera parte, la participante responderá las siguientes preguntas en un apartado distinto en su *Notebook LineaMujeres* bajo el nombre "Interpretación de datos A5":

- ¿Qué ocupación tienen las mujeres que más llaman? ¿A qué crees que se deba que las mujeres con esta ocupación llamen más?
- ¿Qué ocupación tienen las mujeres que menos llaman? ¿A qué crees que se deba que las mujeres con esta ocupación llamen menos?
- ¿Cuál de las dos gráficas de barras crees que sirve más para representar la cantidad de llamadas por ocupación de las mujeres? ¿Por qué?
- ¿Qué tipo de servicio se solicita más? ¿Relacionas esto con algún problema general de las mujeres?
- ¿Qué situaciones crees que influyen en que las mujeres con ocupación HOGAR llamen por VIOLENCIA FAMILIAR?
- ¿Qué tipo de servicio crees que solicitan más las mujeres de la edad en que más llamadas hacen? ¿Por qué ese servicio?

Notas para apoyar la actividad:

- La documentación de los pasos a realizar en la primera parte de la actividad, así como del uso y parámetros de las funciones, deberá llevarse a cabo en su *notebook LineaMujeres* en un nuevo apartado con el subtítulo "Ciencia de datos actividad 5".

Actividad 6: Condicionando mi análisis.

<table>
<tr><td colspan="2">Aprendizaje esperado: Identificará propiedades de una columna condicionada en otra columna, continuar explorando la visualización de los datos con gráficos de histogramas y datos numéricos.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Notebook:
LineaMujeres
● Notebook:
CuadernoDeEstadistica
● Python v.3.x.x.</td><td>Evidencia:
1. Notebook:
CuadernoDeEstadistica con los apartados Operadores y Gráficas y estadística.
2. Anexos MPT3A5_Anexos.pdf resueltos.

Producto:
1. Notebook:
LineaMujeres con los apartados Ciencia de datos actividad 6 e Interpretación de datos A6.</td><td>Retroalimentación:
El apartado Conceptos del notebook CuadernoDeEstadística, deberá contar con la documentación de la siguiente información:
● < (menor que)
● > (mayor que)
● <= (menor o igual que)
● >= (mayor o igual que)
● == (igual)
● != (diferente de)
● & (and)
● | (or)
El apartado Operadores y Gráficas del notebook deberá contar con las respuestas correspondientes a las preguntas sobre histogramas de la primera parte de la actividad.

Evaluación:
● Para las preguntas planteadas en la segunda parte de esta actividad para el dataframe del notebook LineaMujeres deberán responderse de acuerdo a la</td></tr>
</table>

		observación de los datos que se trabajan, además de trabajar las líneas de código de acuerdo a lo que se solicita. ● Dentro del notebook *LineaMujeres* también deberá existir el apartado *Interpretación de datos A6*, que deberá contener las respuestas a las preguntas de la tercera parte de la actividad.

Desarrollo de la actividad:

Primera parte: Operadores lógicos e histogramas.
Sugerencia de tiempo invertido: 30 minutos.

La participante realizará una búsqueda de los operadores relacionales y lógicos que le serán de ayuda para realizar las consultas al *dataframe* del *notebook LineaMujeres*. Por lo anterior, retomará su notebook *CuadernoDeEstadistica*, y en él agregará un apartado con el subtítulo “Operadores”. En dicho apartado, escribirá la definición de cada uno de los siguientes operadores, junto con un ejemplo de su uso:

- < (menor que)
- > (mayor que)
- <= (menor o igual que)
- >= (mayor o igual que)
- == (igual)
- != (diferente de)
- & (and)
- | (or)

Posteriormente en su mismo *CuadernoDeEstadística*, agregará en su apartado *Gráficas y estadística* creado en la actividad anterior, la siguiente información:

- ¿Qué son los histogramas?

- ¿Cuáles son sus características?
- ¿Cómo graficar histogramas utilizando *matplotlib*?

Segunda parte: Relacionando datos.
Sugerencia de tiempo invertido: 2 horas.

La participante manipulará y relacionará diferentes datos del *dataframe* de su *notebook LineaMujeres*. Para ello, se le solicitará realizar los puntos enlistados a continuación, haciendo uso de los operadores lógicos que buscó en la primera parte de esta actividad. Ella deberá escribir el código que crea necesario para poder obtener la respuesta a las siguientes preguntas, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se le motivará para buscar en internet las soluciones pertinentes a sus dudas, apoyarse en los conocimientos adquiridos hasta este momento, o bien utilizar la estrategia de prueba y error hasta que logre completar los pasos. El código a realizar deberá colocarlo en una celda por pregunta en su *notebook LineaMujeres.*

1. ¿Cuántos registros de llamadas existen donde la "Temática 1" sea "Familiar"?
2. A partir del *dataframe* generado en el punto uno, ¿Cuáles registros de la "Temática 2" se repiten más, dada la "Temática 1"? Mostrará al menos los primeros diez.
3. Con ayuda de la biblioteca *matplolib*, graficará los resultados del punto anterior en una gráfica de barras y en una de pastel.
4. ¿Cuál es la escolaridad que más llamadas por “Temática 1” tiene?
5. ¿Cuáles son los valores más comunes para las usuarias entre la edad de 24 y 35, y la Temática _1 “Penal”?
6. ¿Cuántos registros de llamadas existen para la EDAD mayor a 40 años?
7. ¿Cuántos registros de llamadas existen para la EDAD menor a 25 años? ¿Por qué crees que existen menos registros para este rango?
8. Revisará cuáles son las edades que más se repiten en las llamadas, y responderá lo siguiente: ¿Por qué crees que esas edades se repiten más? ¿A qué lo asociarías?
9. ¿Cuántos registros de llamadas existen donde la "Temática 1" sea "Familiar" y la edad sea mayor a 40 años? (operador a utilizar: *and*).
10. ¿Cuántos registros de llamadas existen donde la "Temática 1" sea "Familiar" o la edad sea mayor a 40 años? (operador a utilizar: *or*).

11. Revisará el tamaño del *dataFrame* para saber la cantidad de registros de llamadas con la función *len()* y explicará porque al sumar el resultado del número de registros del punto nueve y diez, no es igual a la cantidad de registros que existen en el archivo.
12. Se le pedirá a la participante que explique y compare lo que hacen las siguientes dos líneas de código, asumiendo que "es_familiar" corresponde a la consulta que se realizó en el punto número uno y "es_mayor_40" a la consulta del punto número cuatro:

```
df[es_familiar | es_mayor_40][['EDAD','TEMATICA_1']]
df[es_familiar & es_mayor_40][['EDAD','TEMATICA_1']]
```

13. Realizará una gráfica de tipo histograma utilizando la biblioteca *matplotlib* para visualizar las edades de las personas que han llamado a la Linea Mujeres, y que la “Temática 1” sea “Familiar” y responderá: ¿A qué asocias que predomina más una edad en específico?
14. Probará con diferentes *bins* en el histograma y responderá: ¿A qué se debe que el histograma cambia cuando se cambia el número de *bins*? ¿Cuál es la finalidad de este parámetro?
15. Utilizará la función *describe()* sobre la serie y ayudándose de los apuntes de su *CuadernoDeEstadistica* explicará los valores que resultan al utilizar esta función respecto a la gráfica del histograma.
16. Se le pedirá a la participante que explique y compare lo que hacen las siguientes dos líneas de código, y cómo se relacionan los valores resultantes de la función *describe()* con el histograma mostrado en la primera línea de código.

```
df['EDAD'].plot(kind="hist",bins=20)
df['EDAD'].describe()
```

17. Se le pedirá a la participante que explique y compare lo que hacen las siguientes dos líneas de código, y cómo se relacionan los valores resultantes de la función *describe()* con el histograma mostrado en la primera línea de código.

```
df[df['TEMATICA_1']=='PENAL']['EDAD'].plot(kind="hist",bins=30)
df[df['TEMATICA_1']=='CONFLICTOS EMOCIONALES']['EDAD'].describe()
```

Una vez realizado el código para todos los puntos y habiendo contestado las preguntas, la participante deberá documentar en su *notebook* de *LineaMujeres* la finalidad de cada operador lógico que empleó, así como sus parámetros correspondientes (se recomienda hacer uso de los anexos *MPT3A6_Anexos.pdf*).

Tercera parte: ¿Qué me dicen las gráficas?
Sugerencia de tiempo invertido: 30 minutos.

Con base en los datos graficados en la segunda parte, la participante responderá las siguientes preguntas en un apartado distinto en su *notebook LineaMujeres* bajo el nombre "Interpretación de datos A6":

- ¿Qué servicio crees que solicitan más las mujeres que están entre las edades en que más llaman? ¿Por qué crees que las mujeres entre esas edades solicitarían más ese servicio?
- ¿En qué rango de edad estás tú, entre las mujeres que más llaman, o las que menos llaman? ¿Por qué motivo llamarías tú?
- ¿Qué características crees que tienen las mujeres que llaman por la temática "familiar"?
- ¿Qué factores crees que podrían influir para que las mujeres llamen por temática "penal"?
- ¿Hay algún dato que te haya intrigado o interesado al observar las gráficas que realizaste? ¿Cuál? Explica en un párrafo la razón.

Notas para apoyar la actividad:

- La documentación de los pasos a realizar en la primera parte de la actividad, así como del uso y parámetros de las funciones, deberá llevarse a cabo en su *notebook LineaMujeres* en un nuevo apartado con el subtítulo "Ciencia de datos actividad 6".

Actividad 7: Aprendo a dominar el tiempo.

<table>
<tr><td colspan="2">Aprendizaje esperado: Manipular datos de tipo tiempo (datetime) con la biblioteca de python correspondiente y crear visualizaciones.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>● JupyterNotebook CuadernoDeEstadística
● JupiterNotebook LineaMujeres</td><td>Evidencia:
1. JupyterNotebook MisBibliotecas con apartado El tiempo en código
2. JupyterNotebook CuadernoDeEstadistica con apartado de Series de Tiempo añadido.

Producto:
1. JupyterNotebook LineaMujeres con ejercicios de programación y estadística.</td><td>Retroalimentación:
● MisBibliotecas deberá contener el apartado solicitado y abarcar biblioteca datetime, función isocalendar y función weekday de manera completa.
● CuadernoDeEstadística deberá contener el apartado solicitado y las preguntas de manera completa.

Evaluación:
● Revisar que los ejercicios de programación en el Notebook LineaMujeres se ejecuten con lo solicitado y sin generar errores.
● Se deberá atender cada punto de manera completa.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Registro de la Historia.
Sugerencia de tiempo invertido: 30 minutos.

1. La participante realizará una búsqueda de la biblioteca datetime y sus funciones weekday e isocalendar. Inicialmente deberá crear un archivo Jupyter Notebook de nombre "MisBibliotecas", luego, creará un apartado de nombre "El tiempo en código" y, en una celda, plasmará la utilidad de la biblioteca datetime; lo mismo hará con las funciones weekday e isocalendar, en</td></tr>
</table>

celdas diferentes. Este documento funcionará como material de consulta en el futuro, o bien, para trabajar los ejercicios próximos.

2. Una vez realizado el primer punto, la participante deberá responder a las siguientes preguntas en el *notebook CuadernoDeEstadistica* en un nuevo apartado de nombre "Series de tiempo".
    - ¿Cuál es la diferencia entre *dataframe* y *series*? En esta pregunta, se deberá resaltar a la participante que ya ha trabajado con ello anteriormente. La respuesta se escribirá tanto en su *notebook CuadernoDeEstadistica*, así como en el *notebook MisBibliotecas* dentro del apartado *El tiempo en código.*
    - ¿Qué son las series de tiempo?
    - ¿Para qué se usan las series de tiempo?
    - Da tres ejemplos de lo que se puede analizar con una serie temporal.

Segunda parte: Las fechas en mi *dataframe*.
Sugerencia de tiempo invertido: 2 horas.

La participante manipulará y relacionará diferentes datos del *dataframe* en su *notebook LineaMujeres*. Para ello, se le solicitará realizar los puntos enlistados a continuación. Ella deberá escribir el código que crea necesario para poder obtener la respuesta a las siguientes preguntas, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se motivará a la participante para buscar en internet las soluciones pertinentes a sus dudas, apoyarse en los conocimientos adquiridos hasta este momento, o bien utilizar la estrategia de prueba y error hasta que logre completar los pasos. El código a realizar deberá colocarlo en una celda por punto en su notebook *LineaMujeres*.

1. Visualizará la columna "FECHA_HORA_ALTA" como serie.
2. Utilizará la función *to_datetime()* sobre la serie y describirá el resultado que se muestra con respecto al de la celda anterior.
3. ¿En qué fechas hubo más llamadas? y ¿En qué fechas hubo menos llamadas?
4. Visualizará en una gráfica una serie de tiempo que muestre la cantidad de registros de las llamadas con las fechas.
5. Visualizará en una gráfica una serie de tiempo que muestre la cantidad de registros de llamadas y las fechas de las llamadas en cada año, mostrando la variación de cada año con un color diferente.
6. Importará la biblioteca *datetime.*
7. Escribirá la siguiente línea de código y describirá qué hace, apoyándose de la búsqueda de la primer parte de la actividad:

    df["DIA_DE_SEMANA"]=df['FECHA_HORA_ALTA'].apply(datetime.datetime.weekday)
8. Visualizará en una gráfica de barras la cantidad de llamadas que se recibieron cada día de la semana.

9. Apoyándose del resultado del punto anterior responderá lo siguiente: ¿Qué día de la semana recibe más llamadas? ¿Qué día de la semana recibe menos llamadas? ¿A qué asociarías que en cierto día específico se reciben más o menos llamadas?
10. Visualizará en una gráfica de barras la cantidad de llamadas por día de la semana y por año con un color distinto para cada año, se sugiere crear un nuevo *dataframe* a partir del principal para almacenar los datos que se piden y luego graficarlos.
11. A partir del punto anterior responderá lo siguiente: ¿En qué día y de qué año se recibieron más llamadas? ¿En qué día y de qué año se recibieron menos llamadas?
12. Visualizará en una gráfica las llamadas de la semana condicionando las temáticas que más le llame la atención para diferentes años.
13. Identificará el número de la semana de las fechas y graficará el número de llamadas por el día de la semana.

Tercera parte: Cotidianamente… ¿Quién hace qué?
Sugerencia de tiempo invertido: 30 minutos.

De acuerdo al Sistema de Indicadores de Género del INMUJERES, en 2010 las mujeres destinaron 46.7 horas de trabajo a la semana, mientras que los varones 41.8 de horas en total, habiendo una diferencia de 4.9 horas, siendo el trabajo doméstico y de cuidado realizado mayormente por mujeres. Si se considerara en términos de economía, en datos del 2009 equivaldría al 21.6% del PIB (INEGI, 2012). El trabajo doméstico y de cuidado no remunerados, afectan directamente a las mujeres, ya que se genera un obstáculo para la igualdad de oportunidades al no contar con el mismo tiempo de recreación que los hombres.
Dicho lo anterior, en este ejercicio, la participante leerá cuidadosamente cada una de las actividades desglosadas en el siguiente recuadro, y a continuación marcará o tachará la casilla de la persona que se dedica principalmente a dicha actividad en el día a día, se sugerirá nombrar a todas las personas con las que habita. Finalmente deberá sumar todos los puntos acumulados por integrante, en el apartado *Total* al final del cuadro. En caso de que la participante no habite con nadie, podrá responder de acuerdo a cómo era cuando habitaba con sus tutores o familiares.
Ejemplo:

Actividades cotidianas	La participante	Mamá	Papá	Hermano	Otro ¿quién?
¿Quién suele lavar los trastes después de cada comida?					

	¿Quién cocina la mayor parte del tiempo?					
	Cuando alguien se enferma, especialmente ancianos o niños, ¿quién hace las labores de cuidados?					
	¿Quién es la persona que suele llevar el sustento económico de la casa?					
	Si hay o hubiera infantes en casa, ¿Quién tiene mayor participación en la crianza?					
	Si se necesita algo para cocinar o para cualquier aspecto del hogar, ¿Quién sale a comprarlo la mayoría del tiempo?					
	Si algo está sucio ¿Quién toma la iniciativa de limpiarlo?					
	Cuando una tubería se rompe, o hay un desperfecto en la electricidad o en el hogar en general, ¿Quién suele arreglarlo?					
	Total					

Una vez resuelto el cuadro anterior, se le proporcionará a la participante las siguientes preguntas para responderlas:

1. ¿Consideras que en el caso de las actividades domésticas, las mujeres tienen una mayor carga? ¿A qué crees que se deba?
2. Si durante el día dedicaras el 70% de tu tiempo a actividades domésticas, ¿en qué invertirías el 30% restante?, ¿consideras que sería tiempo suficiente para hacer todas las actividades que te gustarían?
3. ¿Harías trabajo o trabajarías en un lugar donde no te pagan? ¿Por qué?
4. ¿Por qué crees que las actividades domésticas y de cuidado las "deben" asumir la mayoría de las mujeres?

Referencias:

- INEG.I (Sin fecha). *Trabajo* http://estadistica.inmujeres.gob.mx/myhpdf/140.pdf (noviembre, 2020).

Actividad 8: Igual pero diferente.

<table>
<tr><td colspan="2">Aprendizaje esperado: Manipular las fechas y cuentas utilizando índices, grupos y agregación.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Retroalimentación/Evaluación</td></tr>
<tr><td>● Notebook: LineaMujeres
● Notebook: MisBibliotecas
● Notebook: CuadernoDeEstadistica.</td><td>Evidencia:
1. Notebook: MisBibliotecas con apartado Funciones.
2. Notebook: CuadernoDeEstadistica con el apartado Mi experiencia con la estadística.

Producto:
1. Notebook: LineaMujeres con ejercicios de programación y estadística.</td><td>Retroalimentación:
● MisBibliotecas deberá contener el apartado solicitado y abarcar Índices, función de agrupamiento (group_by) y función de agregación tipo suma (sum) de manera completa.
● Las preguntas que contendrá CuadernoDeEstadística en el apartado Mi experiencia con la estadística serán de respuesta libre.

Evaluación:
● Revisar que los ejercicios de programación en el Notebook LineaMujeres ejecuten lo que se solicita y sin generar errores.
● Se deberá atender cada punto de manera completa.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Haciendo espacio para una nueva versión.
Sugerencia de tiempo invertido: 30 minutos.</td></tr>
</table>

1. La participante retomará el *Notebook MisBibliotecas,* y realizará una búsqueda de información acerca de los Índices (*index),* función de agrupamiento (*group_by)* y función de agregación de tipo suma (*sum)* en Pandas. Deberá especificar la definición, la utilidad de cada función y sus parámetros.
2. En una celda creará un nuevo apartado de nombre "Funciones", en otra celda colocará qué es un índice y su utilidad*,* en otra celda, colocará la función de agrupamiento (*group_by)* y en otra celda plasmará la función de agregación de tipo suma *(sum)*. Este documento funcionará como material de consulta en el futuro, o bien, para trabajar los ejercicios próximos.

Segunda parte: Más funciones para visualizar los datos.
Sugerencia de tiempo invertido: 2 horas.

La participante manipulará y relacionará diferentes datos del *dataframe* en su Notebook *LineaMujeres*. Para ello, se le solicitará realizar los puntos enlistados a continuación. Ella deberá escribir el código que crea necesario para poder obtener la respuesta a las siguientes preguntas, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se motivará a la participante para buscar en internet las soluciones pertinentes a sus dudas, apoyarse en los conocimientos adquiridos hasta este momento, o bien utilizar la estrategia de prueba y error hasta que logre completar los pasos. El código a realizar deberá colocarlo en una celda por punto en su notebook *LineaMujeres*.

1. En la celda donde se lee el archivo con Pandas, la participante deberá agregar el parámetro *parse_dates*, para que Pandas pueda reconocer la columna "FECHA_HORA_ALTA" como fechas.
2. ¿Cual es el índice de la tabla?
3. Creará un *dataFrame* "eventos" a partir de los datos de la tabla, con dos columnas, una de las fechas en las que se recibieron las llamadas, y otra para el número de llamadas en esa fecha, el índice deberá ser la fecha.
4. Graficará como serie de tiempo el punto anterior.
5. Al *dataframe eventos* se le agregará una columna del año de la fecha de las llamadas. (AÑO_ALTA)
6. Visualizará en una gráfica como serie de tiempo la cantidad de registros de llamadas y las fechas de las llamadas en cada año (AÑO_ALTA), mostrando la variación de cada año con un color diferente.
7. Al *dataframe eventos* se le agregará una columna del día de la semana de la fecha de las llamadas (DIA_DE_SEMANA).
8. Utilizará la función *group_by* para visualizar los datos del *dataframe* por los "DIAS_DE_SEMANA" y para la columna "NUMERO_LLAMADAS" utilizará las funciones *aggregate* y *sum* para visualizar la cantidad de llamadas por días de la semana.
9. Visualizará los datos del punto ocho en una barra de pastel.

10. Utilizará la función *group_by* para visualizar los datos del *dataframe* por "AÑO_ALTA y DIAS_DE_SEMANA" y para la columna "NUMERO_LLAMADAS" utilizará las funciones *aggregate* y *sum* para visualizar la cantidad de llamadas por días de la semana y año.
11. Con el último *dataframe* obtenido, realizará un gráfico del número de llamadas por semana y por año.
12. Utilizando la misma estrategia iniciada desde el punto tres, visualizará en una gráfica las cuentas de la semana condicionado en la "Temática_1" FAMILIAR para el año 2019. Probará condicionando diferentes valores para la temática 1 para diferentes años.
13. Utilizando la misma estrategia iniciada desde el punto tres, identificará el número de la semana de las fechas y graficará el número de llamadas por el número de la semana.

Para finalizar responderá:

- ¿Lograste completar los puntos pedidos en esta parte de la actividad?
- ¿Se te hizo más fácil completarlos utilizando las funciones y la estrategia mostrada?
- ¿Crees que sería más difícil realizarlos con los conocimientos adquiridos hasta antes de esta actividad?

Tercera parte: La magia de la estadística.
Sugerencia de tiempo invertido: 30 minutos.

A modo de reflexión acerca de los conocimientos adquiridos de estadística y su conjunción con la programación, responderá las siguientes preguntas en el *Notebook CuadernoDeEstadistica* en un apartado de título "Mi experiencia con la estadística".

- ¿Habías tenido un acercamiento a la estadística anteriormente?
- De acuerdo a tu experiencia hasta ahora, por medio de la estadística, ¿cómo crees que podría ayudar la manipulación y análisis de la información a tu comunidad?
- ¿Se te ha dificultado aprender sobre conceptos estadísticos? ¿Por qué?
- ¿Qué es lo que ha atrapado tu atención al tener acceso a grandes grupos de información?
- ¿Para qué crees que te servirá la estadística descriptiva? En caso de no estar familiarizada con el concepto, se sugerirá hacer una breve búsqueda para responder esta pregunta.

Al finalizar, comentarán las respuestas y conclusiones de participante a participante, o de participante a tallerista.

Actividad 9: Limpio los datos.

<table>
<tr><td colspan="2">Aprendizaje esperado: Identificar datos que no necesariamente estén correctos y lidiar con ellos a través de filtrarlos, asignarles valor por default o eliminar de versiones futuras.</td><td>Duración de la actividad: 3 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>● Archivo Línea mujeres.</td><td>Producto:
1. Dataframe dataClean.csv en archivo LineaMujeres.</td><td>Evaluación:
● El dataframe deberá cumplir con los requisitos que se presentan en la tercera parte de la actividad.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: ¿Quién se equivocó?
Sugerencia de tiempo invertido: 30 minutos.

La participante visualizará el dataframe completo de su notebook LineaMujeres, el nombre de las columnas y los valores que éstas toman en diferentes registros. Posteriormente contestará las siguientes preguntas:
1. ¿Qué columnas tienen valores numéricos y qué columnas tienen valores de cadenas?
2. ¿Se leen igual los valores numéricos que los valores de cadenas?
3. ¿Se pueden interpretar los valores numéricos y los valores de cadena?
(Ver apartado de notas).
4. ¿Qué error presenta la columna FECHA_HORA_ALTA?
5. ¿Crees que esos datos puedan ser útiles para sacar conclusiones sobre su significado? ¿Por qué?
6. ¿Puedes identificar qué columnas del dataframe presentan el mismo dato pero de distinta forma?
7. ¿Qué es la redundancia de datos?
8. ¿Cómo puede corregirse o evitarse?
Se motivará al diálogo con la participante para responder estas preguntas y poder hacer observaciones en caso de que no cuente con un concepto claro de las problemáticas o inconvenientes que presenta la aparición de datos erróneos, sin valor o incoherentes al momento de analizar situaciones a través del filtrado de datos recabados por alguna instancia.</td></tr>
</table>

Segunda parte: Errar es de datos.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Con ayuda de una búsqueda en internet y por medio de la aplicación al *dataframe* de las funciones que se presentan a continuación, la participante responderá las siguientes preguntas. Ella deberá seleccionar la función que crea necesaria para encontrar la solución a cada pregunta, por lo que es importante resaltar que no se le brindará ayuda durante dicho procedimiento. En su lugar, se le motivará para apoyarse en la estrategia de prueba y error hasta que logre completar todo el ejercicio.

Lista de funciones:

- *Unique*
- *np.product*
- *np.nan*
- *Isnull*
- *Notnull*
- *dropna*

Preguntas:

1. ¿Qué tipos de OCUPACIÓN se registran y cuántos son?
2. ¿Qué tipos de temáticas se registraron en el campo TEMATICA_1 y ¿Cuántas son?
3. Listar todos los registros que tienen NULL como TEMATICA_1 y ¿Cuántas son?.
4. Listar todos los registros que tiene una temática registrada en TEMATICA_1. y ¿Cuántas son?
5. ¿La suma de los dos puntos anteriores dan como resultado la cantidad de registros en el *dataframe*?
6. Eliminar todos los valores nulos registrados para la columna CP_HECHOS y aplicar la operación necesaria para saber de qué código postal se registran más llamadas ¿A qué alcaldía pertenece este código postal?

Una vez respondidas todas las preguntas, la participante deberá documentar en su *notebook* de *LineaMujeres* la utilidad de cada función, así como sus parámetros correspondientes (Ver apartado de notas).

Tercera parte: Limpio y reestructuro datos.
Sugerencia de tiempo invertido: 1 hora.

Con todos los conocimientos aprendidos durante esta actividad, la participante deberá realizar la limpieza de los datos que no sirvan del *dataframe LineaMujeres*. Deberá buscar la forma más óptima de hacerlo para cumplir con los siguientes requerimientos:

- El *dataframe* no deberá contener valores nulos en columnas.
- Todas las temáticas deberán contar con un tipo de temática registrada.
- La columna FECHA_HORA_ALTA no deberá incluirse.

Todo lo realizado para este punto, deberá ser documentado con *Markdown* en el mismo *notebook LineaMujeres*.

Por último, la participante investigará cómo guardar un *dataframe* en un archivo csv, de tal forma que el *dataframe* que limpió en esta tercera parte de la actividad, sea guardado en un archivo de nombre “dataClean.csv”.

Para comprobar que el *dataframe* se guardó correctamente, la participante deberá cargar el archivo *dataClean.csv* en el *notebook LineaMujeres* y se deberá visualizar sin problemas.

Notas para apoyar la actividad:

- Para el desarrollo de toda la actividad, se recomienda que la participante cuente con mayor libertad para solucionar los posibles problemas que se le presenten.

Notas (parte 1):

- En caso de desconocer algún valor, la participante realizará una búsqueda en internet para obtener información respecto a su utilidad y significado.

Notas (parte 2):

- Se sugiere que la participante continúe practicando la búsqueda de datos, haciendo uso de las funciones aprendidas en esta parte de la actividad.

Actividad 10: Normalizo los datos.

<table>
<tr><td colspan="2">Aprendizaje esperado: Aplicar técnicas de normalización por escala y por contexto de los datos.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>Notebook CuadernoDeEstadistica.
Notebook LineaMujeres.
Matplotlib de Python.</td><td>Evidencia
Actualización de Notebook CuadernoDeEstadistica
Actualización de Notebook LineaMujeres
Plantillas en MPT3A10_Anexos.pdf resueltas.</td><td>Retroalimentación
Las notas del CuadernoDeEstadística deben corresponder a los puntos.
Se verificará que la secuencia de instrucciones en la notebook sea la correcta inspeccionando la salida producida en el Notebook.
Revisará que las plantillas en MPT3A10_anexos.pdf estén resueltas.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Vamos a dar orden.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante llevará a cabo una búsqueda exhaustiva sobre los siguientes puntos:
¿Qué son los índices en las bases de datos? ¿Para qué sirven?
¿Qué es reindexar datos en una base de datos y cuál es su utilidad?
¿Cómo se reindexan datos en lenguaje Python?
Para la búsqueda deberá apoyarse de diversas fuentes, tanto estadísticas, como de programación, con el objetivo de obtener un aprendizaje sustantivo.
La información encontrada deberá documentarla en su CuadernoDeEstadística, haciendo uso de la sintaxis de Markdown.
Por último, la participante en su notebook LineaMujeres, realizará una lista de los datos que considera sería de ayuda reindexar en el dataframe que ha manejado a lo largo del taller, para lo cual, se recomienda tomar como base las siguientes preguntas:</td></tr>
</table>

- ¿Considero que es necesario reorganizar los datos de mi *dataframe dataClean.csv*? ¿Por qué?
- ¿Qué datos creo necesario reorganizar?

Segunda parte**:** Valores categóricos.
Sugerencia de tiempo invertido: 2 horas y 30 minutos.

Se le solicitará a la participante realizar la siguiente secuencia de instrucciones:

1. Hará uso del anexo *MPT3A10_anexo.pdf* plantilla 1 para comprender el concepto de datos categóricos, su utilidad e importancia dentro del análisis de datos y en los mismos anexos con la plantilla 2, hará una lista de los datos que considere son categóricos en el *dataframe* que ha manipulado a lo largo del taller.
2. Buscará en internet cómo reindexar *dataframes* y *series* en *Python*. Una vez comprendido el uso y la forma de reindexar, creará el código necesario para realizar las siguientes acciones sobre el *notebook LineaMujeres* utilizando el *dataframe* cargado por primera vez. Un punto por celda.
    - Tomando como base el *dataframe* cargado por primera vez en su *notebook*, creará un nuevo *dataframe* que únicamente contenga las columnas correspondientes a las temáticas, desde la 1 hasta la 7. El nuevo *dataframe* contendrá valores nulos en los registros, por lo que se deberá reindexar para que todas las columnas sustituyan esos valores por la leyenda “SIN TEMÁTICA”. Así mismo, se identificará la cantidad de registros para cada valor único de las temáticas y posteriormente, con esos datos numéricos, realizará una gráfica de pastel que ilustre este punto.
    - Creará una serie que contenga únicamente la columna correspondiente al registro de los municipios (columna MUNICIPIO_HECHOS). En caso de que contenga valores nulos, reindexará la serie para que esos valores se sustituyan por la leyenda “SIN REGISTRO”. Así mismo, se identificará la cantidad de registros por valor único en la serie y realizará una gráfica de pastel para ilustrar los resultados.
    - Creará una serie que contenga únicamente la columna correspondiente al registro de los estados de la república de donde llaman (columna ESTADO_HECHOS). En caso de que contenga valores nulos, reindexará la serie para que esos valores se sustituyan por la leyenda “DESCONOCIDO”. Así mismo, se identificará la cantidad de registros por valor único en la serie y realizará una gráfica de pastel para ilustrar los resultados.
    - Creará una serie que contenga únicamente la columna correspondiente al registro de las colonias (columna COLONIA_HECHOS). En caso de que contenga valores nulos, reindexará la serie para que esos valores se sustituyan por la leyenda “DESCONOCIDO”. Así mismo, se identificará la cantidad de registros por valor único en la serie y realizará una gráfica de barras para ilustrar los resultados.

3. Buscará en internet como guardar gráficas con formato de imagen (jpg o png) con ayuda de la biblioteca *matplotlib*. Posteriormente creará una carpeta dentro de la carpeta que contiene su *notebook* y la llamará "gráficas". Utilizará esta carpeta para guardar todas las gráficas creadas a lo largo del taller.
   Para comprobar que se realizó todo de manera correcta, visualizará en su carpeta las imágenes.
4. Con base en las gráficas obtenidas, contestará las siguientes preguntas:
   - ¿De qué municipio se registran más llamadas?
   - ¿De qué estado de la república se registran más llamadas?
   - ¿De qué colonia se registran más llamadas?
   - ¿Por qué existen registros sin temática? ¿Predominan sobre los que sí tienen una temática registrada?

   Las respuestas deberán ser compartidas con la tallerista para abrir un diálogo de retroalimentación sobre la actividad.
   (Ver apartado de notas).

Notas (parte 2):

- En el punto dos de esta parte de la actividad, deberán ordenarse todos los *dataframes* y series por orden alfabético, para ello utilizar la función *sort_values* de pandas.

Links para aprender más:

- IBM konowledge Center. (Sin fecha). *Tipo de variables.* https://www.ibm.com/support/knowledgecenter/es/SSLVMB_sub/statistics_mainhelp_ddita/spss/base/chart_creation_var types.html (noviembre, 2020).
- Pandas. (Sin fecha). *pandas.DataFrame.reindex.* https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.reindex.html (noviembre, 2020).

Actividad 11: Una nueva fuente de datos.

<table>
<tr><td colspan="2">Aprendizaje esperado: Practicar los métodos, técnicas y conceptos aprendidos en un nuevo dataset y usando la plataforma Colab de Google.</td><td>Duración de la actividad: 4 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>1. Cuenta de dominio Gmail para GoogleColab.
2. Hojas de papel, bolígrafo o lápiz.</td><td>Evidencia:
● Preguntas respondidas del punto uno de la primera parte de la actividad.
● Preguntas de reflexión respondidas de la tercera parte de la actividad.
Producto:
● NotebookColab Datos de [nombre del dataset que hayan elegido] con los ejercicios de programación y análisis de datos propuestos en la segunda parte de la actividad.</td><td>Retroalimentación:
● Para las respuestas de la primera parte de la actividad tendrán que expresar las funciones que brinda tanto JupyterNotebook como GoogleColab, así como la identificación de sus diferencias y semejanzas.
● Para las respuestas de las preguntas de reflexión tendrán que ser detalladas.
Evaluación:
● Para el NotebookColab Datos de [nombre del dataset que hayan elegido] deberán cubrir todos los puntos solicitados en la segunda parte de esta actividad y resolverlos por su cuenta empleando los conocimientos que han desarrollado hasta el momento.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: De Júpiter a la Tierra.
Sugerencia de tiempo invertido: 30 minutos.</td></tr>
</table>

Las participantes explorarán la plataforma *GoogleColab* y compararán lo observado con la plataforma *JupyterNotebook* respondiendo las siguientes preguntas:

- ¿Qué similitudes y qué diferencias encuentras entre *JupyterNotebook* y *GoogleColab*? Nombra por lo menos 3 de cada una.
- Revisar memoria disponible y el tiempo que dura una sesión dentro de la plataforma y responder ¿Qué limitaciones crees que podría tener estas condiciones?
- ¿Qué ventajas encuentras al usar la plataforma *GoogleColab*?
- ¿Qué relación hay entre *Jupyt*erNotebook y *GoogleColab*?

Segunda parte: Un nuevo proyecto.
Sugerencia de tiempo invertido: 3 horas.

Las participantes iniciarán un nuevo proyecto con otro *dataset*, para ello se recomienda que utilicen las páginas mostradas en la actividad tres del presente taller:

- Datos Abiertos de México: https://datos.gob.mx/
- Portal de datos de la Ciudad de México: https://datos.cdmx.gob.mx/pages/home/
- Datos generados por la ciudadanía: http://datamx.io/

Se sugerirá el seleccionar datos relacionados con la desigualdad de género, sin embargo, la temática será libre.
Una vez seleccionado el dataset con el que trabajarán, crearán un nuevo *Notebook* en la plataforma Colab de Google, asignándole el nombre "Datos de ..." seguido del nombre de la temática del *dataset* que hayan elegido.
Sobre este set de datos aplicarán todos los conocimientos adquiridos hasta este momento en el taller, por lo que, como se les ha estado pidiendo en las últimas actividades se recomienda no brindar ayuda a la participante para realizar los puntos requeridos, en su lugar se le motivará para buscar las soluciones pertinentes en internet o en cualquier otro medio.
Deberán organizar el código escrito en celdas así como se hizo en el *Notebook LineaMujeres*, además el código deberá ser documentado con Markdown.

1. Importará las bibliotecas vistas a lo largo del taller (*csv, pandas, matplotlib, datetime*).
2. Cargará el archivo *csv* para obtener el *dataframe*.
3. Utilizará las funciones básicas para manipular un dataframe (*len, columns, dtypes, info, head, tail, iloc, loc, describe*).
4. Obtendrá el top de los valores más comunes utilizando la función *value_counts*
5. Deberá visualizar al menos tres gráficas de pastel, tres gráficas de barras y tres histogramas.
6. Deberá analizar los datos de una columna condicionada en otra columna.
7. Deberá analizar los datos condicionados en dos columnas.

8. Deberá hacer uso de los operadores lógicos y relacionales para seleccionar datos.
9. Deberá mostrar al menos dos histogramas como series de tiempo, para ello deberá usar la biblioteca *datetime* con sus correspondientes funciones.
10. Deberá contar eventos en una fecha.
11. Deberá condicionar una columna en relación a una columna tipo fecha.
12. Deberá hacer uso de la función *index* para cambiar los índices.
13. Deberá hacer uso de la función *group_by,* acompañada de funciones de agregación para seleccionar los datos.
14. Deberá crear *dataframes y series* a partir del *dataframe* principal para seleccionar o visualizar datos.

Si el set de datos requiere una limpieza y/o normalización (tomar como referencia las actividades nueve y diez del taller), esta se deberá realizar antes de proceder a realizar los puntos requeridos. Este procedimiento se realizará en un *Notebook* con nombre "Limpieza de datos", y guardará el archivo *csv*, una vez hecho esto, se procederá a realizar los puntos enunciados en esta parte de la actividad.
(Ver apartado de notas).

Tercera parte: Los datos hablan por sí mismos.
Sugerencia de tiempo invertido: 30 minutos.

A modo de reflexión, la participante responderá las siguientes preguntas:

- ¿Qué motivó tu selección para la temática del nuevo *dataset*? Detalla los motivos.
- Al trabajar con los datos disponibles, ¿lo que observaste captó aún más tu atención? ¿Por qué?

A partir de lo que observaste en el análisis de datos,

- ¿Te surgieron otras preguntas o inquietudes relacionadas con el tema?
- Al realizar la actividad, ¿Te causó alguna idea general acerca de lo que “dicen” los conjuntos de datos?
- Describe brevemente tu interpretación del conjunto de datos que revisaste.

Notas (parte 1):

- La participante deberá atender por lo menos los puntos enunciados, sin embargo, se le dará libertad para manipular y visualizar los datos de su *dataframe*, por lo que se podrá agregar más código y este también deberá estar documentado.

Actividad 12: Visitando análisis de datos.

<table>
<tr><td colspan="2">Aprendizaje esperado: Explorar análisis de datos e interpretar resultados de estos.</td><td>Duración de la actividad: 4 horas y 30 minutos.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>● Notebook LineaMujeres.
● Notebook CuadernoDeEstadistica.</td><td>Producto:
1. Gráficas solicitadas
2. Notebook CuadernoDeEstadistica
Evidencia:
1. MPT3A12_anexos resueltos.
2. Dataframe analizointerpreto
3. Interpretaciones solicitadas</td><td>Evaluación:
● Para el NotebookColab Datos de [nombre del dataset que hayan elegido] deberán incluir las gráficas solicitadas, junto con su interpretación.
Retroalimentación:
● La discusión deberá realizarse alrededor de los valores cuantitativos, sin embargo se deberá promover que estos valores se ligen a la experiencia de vida de la participante.
● Se deberá hacer un contraste entre la interpretación provista en la segunda parte y el ejemplo de la tercera parte.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Reflexiono sobre mis reflexiones.
Sugerencia de tiempo invertido: 30 minutos.

La participante realizará interpretaciones y análisis al dataframe con el que ha estado trabajando a lo largo del taller. Para cumplir con lo anterior, se le solicitará en un primer momento, responder las siguientes preguntas (se recomienda utilizar los anexos MPT3A12_Anexos.pdf):
● Al trabajar con los datos de tu dataframe, se te ha solicitado en distintas ocasiones reflexionar acerca de estos y su significado. ¿Te ha sido complicado o fácil realizar las reflexiones? ¿A qué crees que se deba?</td></tr>
</table>

- En tu *notebook CuadernoDeEstadistica* se te solicitó buscar y escribir qué es el análisis e interpretación de los datos estadísticos. Con base en tus respuestas, ¿Identificas alguna diferencia entre "interpretar" y "analizar"? ¿Cuál?
- A lo largo de este taller, ¿crees que has estado interpretando o analizando los datos?

Segunda parte: Analizando ando.
Sugerencia de tiempo invertido: 2 horas.
La participante analizará e interpretará algunos datos del *dataframe* con el que ha trabajado a lo largo del taller, poniendo en práctica los conocimientos técnicos que ha adquirido hasta este momento. Con tal intención, se le solicitará a la participante retomar su *notebook LineaMujeres*, y documentar en un apartado de nombre "Analizo e interpreto" la resolución de la siguiente secuencia de pasos (se sugiere hacer uso de los anexos *MPT3A12_Anexos.pdf)*:

1. Identificará dos variables en su *dataframe* que crea que son importantes. Ejemplo: Estado de la República y Servicio.
2. Creará un nuevo *dataframe* de nombre "analizoInterpreto" con los valores de las variables elegidas, tomando los datos del *dataframe* que ha utilizado a lo largo del taller.
3. Observará su *dataframe* y responderá:
    a) ¿Crees que existe una relación entre ambas variables?
    b) ¿Encuentras algún patrón que se repita? Por ejemplo: en los estados de la república del norte, se solicita más un cierto tipo de servicio.
4. Si no encontró alguna relación, continuará con el paso 6.
5. Si encontró alguna relación entre estas dos variables, describirá las características que identifica de dicha relación.
6. Realizará una gráfica donde se puedan cruzar variables (de barras, histograma), y utilizará como parámetros de los ejes (X Y) las dos variables que eligió.
7. Observará la gráfica y responderá:
    a) ¿La descripción que hiciste en el punto 5 concuerda con lo que observas en la gráfica?
    b) ¿Qué observas en la gráfica?
8. Posteriormente, presentará en un apartado distinto de su *notebook* la gráfica que realizó en el paso 7, y acompañará dicha gráfica con una interpretación de ésta tomando como base lo elaborado en los pasos anteriores.
9. En este mismo apartado, la participante además documentará las funciones que tuvo que aplicar en esta segunda parte para resolver las instrucciones solicitadas.

Tercera parte: A interpretar otros datos.
Sugerencia de tiempo invertido: 2 horas.

La participante interpretará los resultados de otro conjunto diferente de datos y leerá los resultados que se le presenten. Para ello, leerá los siguientes párrafos y con base en la página de datos estadísticos del Instituto Nacional de Mujeres, responderá la secuencia de instrucciones que se presenta (se recomienda hacer uso de los anexos *MPT3A12_Anexos.pdf)*:

Caminos de justicia: de la participación política de las mujeres a la igualdad sustantiva
Uno de los derechos políticos de las mujeres es la participación política y el mundo tiene una deuda pendiente con las mujeres, por ello, diversas sugerencias nacionales e internacionales buscan que en los países se haga posible la participación equilibrada entre mujeres y hombres. En México, este ejercicio de justicia comenzó a politizarse en el ámbito público de toma de decisiones a partir de las Cuotas de Género "para 1996 se fijó un límite de 70% de legisladores un mismo género y en 2007 se estableció que las candidaturas para integrar el poder legislativo debían de integrarse, cuando menos, de un 40% por personas de un mismo sexo". (INE)

Para 2019, ya podemos hablar de Paridad de Género pues el 6 de junio, entró en vigor la reforma de 10 artículos de la Constitución Política de los Estados Unidos Mexicanos que indican que la mitad de los cargos de toma de decisiones en los tres órdenes de gobierno, organismos autónomos, candidaturas de los partidos políticos y elección de representantes en los ayuntamientos con población indígena, debe estar ocupado por el 50% de mujeres y el 50% de hombres.

Sin embargo y como los derechos son progresivos, hoy el movimiento feminista y amplio de mujeres tienen en agenda hacer posible la Igualdad Sustantiva que de acuerdo a la Ley de Igualdad Sustantiva entre Mujeres y Hombres en el Distrito Federal publicada en el año 2007, es "(...) el acceso al mismo trato y oportunidades, para el reconocimiento, goce o ejercicio de los derechos humanos y las libertades fundamentales." (Artículo 5 fracción IV).

Lo anterior, quiere decir que si bien ya se tiene el número de mujeres y hombres al frente es pertinente hacer posible la igualdad de condiciones y trato de las mujeres en estos espacios, además de asegurar el ejercicio pleno del poder con el fin de transformar estos espacios y que las mujeres puedan apropiarlos.
(Ver apartado de notas).

1. Habiendo leído los párrafos anteriores, la participante abrirá el siguiente enlace: http://estadistica.inmujeres.gob.mx/formas/temas_descripcion.php?IDTema=8
2. En él, seleccionará la pestaña “Indicadores” que le abrirá una lista de los diferentes puestos económicos y políticos en los que se analiza el nivel de participación de las mujeres.
3. De dicha lista, seleccionará el indicador que más le interese (por ejemplo: Síndicas y síndicos), y éste le arrojará los datos presentados en una tabla.
4. Observará la tabla, y responderá las siguientes preguntas:
    - ¿Qué variables se presentan en la tabla? (Años, cantidad de mujeres y hombres)
    - ¿Cambia la cantidad de mujeres y hombres conforme aumentan los años? ¿Cómo es este cambio?
    - Con base en los párrafos iniciales, ¿opinarías que estos datos cumplen con la paridad de género?
    - Con base en los párrafos iniciales, ¿opinarías que estos datos cumplen con la igualdad sustantiva?
5. La participante repetirá como mínimo cinco veces los puntos 3 y 4, eligiendo cada vez un indicador distinto.
6. A continuación, se le solicitará abrir nuevamente el enlace proporcionado, y seleccionar la pestaña “Indicadores Básicos”. Con base en esta, y las demás pestañas consultadas, responderá:
    - ¿Los datos son presentados al público en tablas, gráficas o texto?
    - Antes de tener dicha presentación, ¿crees que los datos tuvieron que haber estado en un *dataframe*?
    - ¿Qué funciones de Python crees que se tuvieron que haber aplicado al *dataframe* original para que los datos se presentaran en las tablas?
    - Partiendo de esto, ¿cómo crees que deberías presentar los datos con que has trabajado a lo largo del taller?

Notas para apoyar la actividad:

- Se recomienda que para la primera parte de la actividad, la participante comparta las respuestas a las preguntas para discutirlas por medio del diálogo.
- Para la segunda parte de la actividad, es muy importante hacerle notar a la participante que los datos con los que está trabajando, son en su mayoría datos categóricos, por lo cual no realizará exactamente una correlación de datos, ya que esta únicamente puede desarrollarse con datos numéricos.

Notas (parte 3):

- Se sugiere consultar los links de la sección Links para aprender más, para conocer más acerca del tema de igualdad sustantiva y violencia política contra las mujeres.

Links para aprender más:

- Cornejo, Magdalena. (Sin fecha). *Relación entre variables: explicación.* https://es.coursera.org/lecture/estadistica-aplicada-negocios/relacion-entre-variables-explicacion-YVCDq (noviembre, 2020).
- Olle, J. (Sin fecha). *El secreto para Analizar Datos como un pro que un consultor estadístico nunca te contaría.* https://conceptosclaros.com/como-analizar-datos/ (noviembre, 2020).
- Tecnologías Información. (Sin fecha). *Interpretación de los Datos: Objetivos, Tipos y Beneficios*. https://www.tecnologias-informacion.com/interpretacion-datos.html (noviembre, 2020).
- Yrigoyen, M. (2017). *Historia de los derechos de la mujer.* [video] https://www.youtube.com/watch?v=OFYxVCATo1o&ab_channel=MoyyYrigoyen (noviembre, 2020).
- Lascurain, Sofía y Talamás, Marcela. (2016). *Protocolo para atender la violencia política contra las mujeres.* https://www.te.gob.mx/protocolo_mujeres/media/files/7db6bf44797e749.pdf (noviembre, 2020).
- INE Instituto Nacional Electoral. (2017). *Paridad de género y derechos políticos electorales.* https://www.ine.mx/paridad-de-genero-derechos-politicos/(noviembre, 2020).
- Pérez, Catalina. (Sin fecha). *Cronología del movimiento en pro de la paridad de género.* http://www.tiki-toki.com/timeline/entry/28119/Cronologa-del-movimiento-en-pro-de-la-paridad-de-gnero#vars!date=2013-06-16_07:56:02! (noviembre, 2020).

Actividad 13: Recetas de visualización avanzadas

Aprendizaje esperado: Explorar análisis de datos e interpretar resultados de estos.		**Duración de la actividad:** 4 horas y 30 minutos.
Recursos	**Evidencia/Producto**	**Evaluación/Retroalimentación**
● *Notebook LineaMujeres.* ● *Notebook CuadernoDeEstadistica.*	**Evidencia:** - Gráficas en notebook *LineaMujeres* - *Notebook CuadernoDeEstadistica*	**Retroalimentación:** ● Verificar que las gráficas en la notebook LineMujeres correspondan a los datos y estas se desplieguen de forma correcta.

Desarrollo de la actividad:

Primera parte: Ponle estilo a tus datos graficados.
Sugerencia de tiempo invertido: 2 horas.

Se le solicitará a la participante retomar su *notebook LineaMujeres* y con ayuda del dataframe realizará gráficas y subgráficas.

1. A partir de los datos de servicio, graficar con la función plot().
2. Ocultar la leyenda de plot(), utilizando el parámetro legend=False.
3. Ahora volver a mostrar la leyenda, pero alterar el nombre de las etiquetas del eje x y del eje y usando los parámetros xlabel="etiqueta_x", ylabel="etiqueta_y"
4. Cambiar la escala de la gráfica con los parámetros logy=True. Discutir en qué cambió la apariencia de la gráfica y para qué es útil este tipo de gráficas.
5. Agregar un nuevo componente de datos dependientes y graficar dos ejes y utilizando el parámetro secondary_y=True.
6. Subdividir un plot en varios subplots utilizando los datos del dataframe. Utilizando el parámetro subplots=True.

Segunda parte: Para analizar es mejor agrupar.
Sugerencia de tiempo invertido: 2 horas y 30 minutos.

1. Investigar qué es y para qué sirve un “diagrama de caja” (Boxplot).
2. Con los datos de servicio del dataframe, graficar un diagrama de caja usando la función plot.box(). Redactar un párrafo explicando algunas observaciones a partir de las gráficas.
3. Ahora dibujar el diagrama de caja anterior pero de forma horizontal usando el parámetro vert=False.

4. Con los mismos datos, ahora graficar una función de densidad con la función plot.kde(). Completar el párrafo del inciso 2 con una breve explicación de lo que se observa.
5. De los datos de servicios sume en un arreglo todos los servicios de los 32 estados y nombrela total_servicios. Realice una gráfica mostrando el área de dicha variable con la función plot.area(). Ahora agregar la variable de cualquier estado a esa gráfica de área. Agregar al párrafo del inciso 2 qué significa el área que corresponde entre la gráfica de la variable total_servicios y la del estado seleccionado.
6. Investigar para qué sirven las funciones plot.scatter() y plot.hexbin().
7. A partir del dataframe encontrar como aplicar las funciones plot.scatter() y plot.hexbin() y explicar el resultado.

Links para aprender más:

- Wikipedia La enciclopedia libre. (Sin fecha). *Diagrama de caja.* https://es.wikipedia.org/wiki/Diagrama_de_caja (Noviembre 2020).
    - *Area chart.* https://en.wikipedia.org/wiki/Area_chart (Noviembre 2020).

Actividad 14: De dónde vienen los datos

<table>
<tr><td colspan="2">Aprendizaje esperado: Reflexionar sobre la ética de los datos.</td><td>Duración de la actividad: 2 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr><td>● Hojas de papel, bolígrafo, lápiz o cualquier otro medio para escribir.</td><td>Evidencia:
1. Preguntas respondidas de la primera parte de esta actividad.
2. Preguntas respondidas de la segunda parte de esta actividad.
3. Preguntas respondidas de la tercera parte de esta actividad.</td><td>Retroalimentación:
● Revisar que las respuestas vayan más allá de las respuestas cortas, estimular la reflexión en torno al tema de la ética de los datos apoyándose de los conocimientos que han desarrollado hasta el momento para llegar a conclusiones detalladas.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Desarrollo de la actividad:</td></tr>
<tr><td colspan="3">Primera parte: Información fluctuante.
Sugerencia de tiempo invertido: 30 minutos.

Como introducción, la participante reflexionará sobre el dominio de información personal a partir de contextos cercanos, para esto se le proporcionará los siguientes textos y responderá las preguntas:
Como uña y mugre.
"Mi mejor amiga y yo somos muy unidas, conozco tan bien a María que sabía que le haría muy feliz un cactus de regalo para su cumpleaños, sé cómo animarla cuando está triste, nos contamos de todo".
1. ¿La información nos ayuda a tomar decisiones?¿Por qué?
2. ¿Con qué propósitos usas la información de la gente que conoces?
3. ¿Qué similitud notas entre el acceso a la información de la gente que conoces con el acceso a la información pública?
4. ¿Consideras que poseer información de alguien pudiera causarle algún perjuicio si otra persona se lo propusiera?
¿Como libro abierto? ¡No!
"Mi nombre es público, muchas personas conocen mi edad y otras pocas dónde estudio, sin embargo, hay información sensible que puede vulnerar mi integridad humana si llegara a caer en manos equivocadas, y que por tanto, la prefiero en el anonimato, por ejemplo, enfermedades, finanzas personales, domicilio, etc., aunque si necesito un examen médico o hago un trámite en línea, daría acceso a esa información a grupos específicos y ellos deben asegurar que protegerán mi información personal".</td></tr>
</table>

1. ¿Qué es la anonimización de datos?
2. ¿Por qué crees que es importante para algunas personas mantener su información en el anonimato, o bien, reservada para ciertos casos específicos?

Te quiero contar que…

"Mis amigas y yo compartimos información desde el consentimiento, ella sabe cosas de mí porque cedí mi información personal a ella y viceversa".

1. ¿Qué es el consentimiento y cuál es su importancia al tratarse de información personal?
2. ¿Existe una manera de ejercer el consentimiento al tratar información personal en medios digitales? ¿Cuáles?
3. ¿Qué son los derechos ARCO? Escribe qué significa el acrónimo.
4. ¿En qué te beneficia que puedas ejercer tus derechos ARCO?

Segunda parte: Saber es poder.

Sugerencia de tiempo invertido: 1 hora.

Las participantes harán una lectura del caso de Cambridge Analytica suscitado en 2018 para ejemplificar el mal uso de los datos, para esto, leerán al menos cinco notas del caso y responderán las siguiente preguntas, podrán apoyarse en búsquedas de Internet para responderlas:

1. ¿Qué es Cambridge Analytica?
2. ¿Qué hizo que se viera envuelto en polémica en 2018?
3. ¿Qué otra empresa estuvo involucrada?
4. ¿Con qué propósitos hicieron uso de la información?
5. ¿Por qué generó tanto descontento en la sociedad?
6. ¿Qué tipo de información recolectaron de las personas?
7. ¿Cuáles fueron las consecuencias legales y sociales para estas empresas?
8. Si alguien hiciera mal uso de información personal que le confiaste ¿Cómo reaccionarías?
9. A partir de lo revisado, ¿En qué otros medios se proporciona información de manera inadvertida?
10. ¿Qué ventajas se obtienen cuando se posee todo tipo de información?

Tercera parte: Obrar bien, para no herir.

Sugerencia de tiempo invertido: 30 minutos.

Para finalizar, la participante hará una reflexión acerca de la ética de los datos donde contrastará las dos primeras partes de esta actividad para responder las siguientes preguntas:

- ¿Qué entiendes por ética?
- Si los datos vienen de la recolección de las personas con dispositivos tecnológicos ¿por qué es importante considerar la ética cuando hablamos de posesión y manipulación de datos?
- El caso de Cambridge Analytica es polémico por el mal uso de datos ¿Qué otros casos o situaciones conoces que hacen mal uso de los datos?

Actividad 15: Presentando mi proyecto.

<table>
<tr><td colspan="2">Aprendizaje esperado: Elaboración de reportes y presentación oral.</td><td>Duración de la actividad: 7 horas.</td></tr>
<tr><td>Recursos</td><td>Evidencia/Producto</td><td>Evaluación/Retroalimentación</td></tr>
<tr>
<td>

Hojas de papel.
Lápices, plumas o plumones.
Bloc de notas digital.

</td>
<td>
Evidencia:

Transcripción de reporte.
PresentacionRecursiva.
Fichas de consulta sobre Pitch.

Producto:

Reporte “LineaMujeres”
Presentación digital “presentacionLineaMujeres”
Video “pitchDeElevador”

</td>
<td>
Retroalimentación:

En la transcripción del reporte, las partes deberán estar ordenadas de forma coherente.
La presentación recursiva deberá incluir las partes que la conforman así como su descripción.
Las fichas de consulta tendrán que contener lo que es un Pitch así como sus partes, lo cual le servirá para poder desarrollar su propio Pitch más adelante.

Evaluación:

El reporte “LineaMujeres” sobre su proyecto deberá presentar la siguiente estructura:

Introducción.
Motivación.
Desarrollo.
Conclusiones.

La presentación digital “presentacionLineaMujeres” deberá contener:

La estructura de la presentación de acuerdo con la estructura de su pitch.

</td>
</tr>
</table>

		○ Diapositivas o slides con cantidad de texto moderada. ○ Imágenes. ○ Referencias. ● El video “pitchDeElevador” deberá cumplir los siguientes requisitos: - Duración mínima de 30 segundos y máximo de 2 minutos. - Responder las 6 preguntas planteadas en la tercera parte de la actividad con conectividad de ideas. - Cumplir con los 3 puntos de presentación personal mencionados en la tercera parte.

Desarrollo de la actividad:

Primera parte: Reportando lo que hago.
Sugerencia de tiempo invertido: 3 horas.

La participante identificará las partes de un reporte, por medio de las siguientes instrucciones:

1. Se le solicitará buscar en internet qué es un reporte o informe y cuáles son los elementos que conforman su estructura.
2. Se le proporcionará un ejemplo de reporte, en el cual sus partes (introducción, motivación, desarrollo y conclusiones) estén en desorden.
3. Posteriormente, la participante deberá indicar en el mismo reporte que se le proporcionó las partes de este, para finalmente, transcribir dicho documento con sus partes en el orden correcto.
   (Ver apartado de notas).
4. Una vez que la participante no tenga dudas acerca de la estructura de los reportes o informes, se le solicitará elaborar uno con los resultados y la experiencia del trabajo de análisis de datos que hizo a lo largo del taller con su *dataframe* en *LineaMujeres*.

Segunda parte: Te presento mi proyecto.
Sugerencia de tiempo invertido: 2 horas.
La participante llevará a cabo una presentación digital acerca del trabajo que ha realizado a lo largo del taller, para ello deberá realizar las siguientes acciones:

1. Realizará una búsqueda en internet acerca de las partes de una presentación y lo que contiene cada una, cubriendo los puntos mínimos:
    - Diapositivas o slides.
    - Estructura de una presentación.
    - Imágenes.
    - Referencias.
    - Diferentes plataformas que se pueden utilizar para elaborar presentaciones (sólo software gratuito).
2. Documentará la búsqueda realizada a través de la elaboración de una presentación llamada “presentacionRecursiva”, en donde explicará cada uno de los puntos de una presentación y cómo se deben desarrollar, a la vez que aplicará lo aprendido siguiendo cada uno de los puntos investigados para desarrollar la presentación.
3. Desarrollará una presentación de nombre “presentacionLineaMujeres” en la plataforma que desee, con el objetivo de exponer el trabajo que realizó durante este taller y los resultados que obtuvo. Es importante que la presentación muestre el proceso que se llevó a cabo:
    - Aprendizajes adquiridos.
    - Herramientas utilizadas.
    - Objetivo del taller.
    - Proceso de interpretación de resultados (mostrará algunas gráficas de las que elaboró en el taller).
    - Resultado final.

(Ver apartado de notas).

Tercera parte: 1 minuto es suficiente.
Sugerencia de tiempo invertido: 2 horas.

La participante desarrollará una presentación verbal con la técnica *Pitch de elevador* que consiste en presentar una idea o proyecto de forma llamativa y convincente en cuestión de segundos o máximo 2 minutos, que es el tiempo que tardaría un viaje en ascensor.

1. Para lo anterior, la participante buscará en internet qué es, en qué consiste y cuáles son las partes de un *pitch* (conjuntará la información encontrada en fichas de consulta).
2. Posteriormente, responderá las siguientes preguntas que le ayudarán a formular cada uno de los puntos a exponer en su *pitch* acerca del tema *mis habilidades en ciencia de datos*:
    - Planteamiento de una pregunta para llamar la atención.
    - Presentación: ¿Quién eres? ¿A qué te dedicas?
    - ¿Qué problemática resuelve tu proyecto?
    - ¿Qué solución planteas?
    - ¿Qué beneficios se obtienen de tu trabajo?
    - ¿Por qué eres la persona adecuada para resolver el problema?

3. Una vez respondidas las preguntas, la participante deberá consultar videos en internet que muestren ejemplos de un *pitch* de elevador. Con base en lo observado, deberá redactar un texto a manera de discurso que una todas las respuestas a las preguntas del punto 2.
4. Posteriormente lo leerá en voz alta, si es posible a otras participantes, a la tallerista, o bien a alguna persona conocida. En caso de ser necesario, se podrán modificar las respuestas de las preguntas las veces que sean necesarias para lograr un resultado favorable.
5. Cuando las respuestas sean definitivas y la participante quede conforme con el texto desarrollado, pasará a trabajar en su discurso oral. Para ello, expondrá su texto varias veces frente a una persona o frente a un espejo, poniendo especial atención en los siguientes elementos:
    - Tono y volumen de voz.
    - Gesticulación y dicción.
    - Lenguaje corporal: postura, dirección de la mirada, posición de las manos, expresión facial, desplazamiento en el espacio.

    Todos estos elementos le serán de utilidad para establecer una conexión con el público que la escucha y generar su interés por el contenido de su discurso.

Una vez que la participante se sienta convencida de su discurso, grabará un video realizando su *pitch* tomando en cuenta todos los aspectos trabajados durante el desarrollo del mismo. Este video deberá contar con los siguientes requisitos:

- Deberá aparecer la participante en primer plano del video.

- Mostrará la presentación de su proyecto elaborado en la segunda parte de esta actividad al mismo tiempo que presenta su *pitch*.

Este video se hará público en las redes sociales del PILARES que le corresponda, para que pueda estar disponible a todas las personas interesadas.
(Ver apartado de notas).

Notas para apoyar la actividad:

- Se recomienda que la participante ponga en práctica distintos ejercicios que le ayudarán a tener un mejor manejo de su lenguaje corporal y dicción al momento de exponer su pitch. Alguno de estos es: colocar un lápiz entre los labios e intentar pronunciar lo más comprensible que se pueda una frase.

Notas (parte 1):

- Se recomienda que para facilitar este ejercicio, las partes del reporte de ejemplo estén redactadas de forma precisa, concreta y clara, para que al momento de reorganizarlas, la participante pueda identificar cada una de estas sin problema.

Notas (parte 2:

- La presentación del punto tres deberá contener como mínimo 10 slides en los que deberá aplicar buenas técnicas de desarrollo de una presentación. En caso de que la participante las desconozca, deberá realizar una búsqueda de estas.

Notas (parte3):

- Se sugiere consultar los videos anexados a la sección de Links para aprender más, para ver ejemplos de pitch de elevador.

Links para aprender más:

- Ro, Alicia. (2018). *Ejemplo de Elevator Pitch para presentarte tú o un proyecto.* [video] https://www.youtube.com/watch?v=uv357YzY7-k (Noviembre, 2020).
- iurisdocTV. (2013). *Elevator pitch. Tienes 20 segundos*-eduCaixa [video]

- https://www.youtube.com/watch?v=2b3xG_YjgvI (Noviembre, 2020).ThinkBigJovenes. (2014). *Elevator Pitch-HuertoApp-Eva Fernández.* [video] https://www.youtube.com/watch?v=Y7ClvtxcJ64 (Noviembre, 2020).