

# Informe de Laboratorio 04

## Tema: Python

Nota

Estudiante	Escuela	Asignatura
Gonzalo Rail Layme Mamani glaymem@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación web Semestre: III Código: 20231001

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 2 Junio 2023	Al 6 Junio 2023

## 1. Tarea

- General: Diseña responsablemente aplicaciones web, sus componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
- Específica: Construye responsablemente soluciones con tecnología web siguiendo un proceso adecuado llevando a cabo las pruebas ajustada a los recursos disponibles del cliente.
- Específica: C.p. Aplica de forma flexible técnicas, métodos, principios, normas, estándares y herramientas del desarrollo web necesarias para la construcción de aplicaciones web e implementación de estos sistemas en una organización.

## 2. Resultados del estudiante

- RE. 2: La capacidad de aplicar diseño de ingeniería para producir soluciones a problemas y diseñar sistemas, componentes o procesos para satisfacer necesidades específicas dentro de consideraciones realistas en los aspectos de salud pública, seguridad y bienestar; factores globales, culturales, sociales, económicos y ambientales.
- RE. 8: La capacidad de crear, seleccionar y utilizar técnicas, habilidades, recursos y herramientas modernas de ingeniería y tecnologías de la información, incluyendo la predicción y el modelamiento, con una comprensión de las limitaciones.

### 3. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Algoritmo de ordenamiento por inserción

### 4. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/rescobedoq/pw2.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- [https://github.com/GonzaloRail/Laboratorio\\_04\\_PWEB](https://github.com/GonzaloRail/Laboratorio_04_PWEB)

## 5. Marco Teórico

### 5.1. Python

- Python es un lenguaje de programación interpretado no fuertemente tipado.

### 5.2. Instalación en GNU/Linux

- Para instalar Python 3 en cualquier distribución GNU/Linux use sus mismos repositorios.
- Por ejemplo en sistemas operativos compatibles con GNU/Linux Debian use alguno de los dos comandos siguientes:

Listing 1: Instalar Python en GNU/Linux Debian

```
# apt-get install python3
```

Listing 2: Instalar Python en GNU/Linux Ubuntu

```
$ sudo apt-get install python3
```

### 5.3. Instalación en MS Windows

- Para descargar Python compatible con sistemas operativos MS Windows utilice alguno de los instaladores de: <https://www.python.org/downloads/windows/>
- Descargar e instalar en sistemas operativos MS Windows, por lo general es muy sencillo. Además si usted es un usuario nativo de windows este proceso será casi intuitivo. No tenga miedo de instalar estos programas todo es software libre, así que no necesitará parches o cracks. :)

## 5.4. Instalación en MS MacOS

- Para instalar Python 3 en sistemas operativos MacOS puede descargar el instalador desde: <https://www.python.org/downloads/macos/> o usar brew:

Listing 3: Instalar Python en MacOS

```
$ brew install python
```

## 5.5. Comprobar presencia y versión de Python

- Debe comprobar que la instalación y el reconocimiento del compilador ya estan presentes en su sistema operativo:

Listing 4: Verificando presencia y versión de Python

```
$ python3 --version
```

- Usted debería obtener como salida la versión que tiene instalada.

Listing 5: Verificando presencia y versión de Python

```
Python 3.11.2
```

## 5.6. Hola mundo

- Cree su primer ejercicio **helloworld.py**

Listing 6: Creando el archivo helloworld.py

```
$ vim helloworld.py
```

Listing 7: helloworld.py

```
print("Hello World!")
```

- Para ejecutar su script **helloworld.py** ejecute el siguiente comando:

Listing 8: Ejecutando el script helloworld.py

```
$ python3 helloworld.py
```

## 5.7. Comentarios en Python

- Los comentarios de Python solo se aplican por cada línea.
- Pero usted puede utilizar varias técnicas para comentar en el editor de Vim

Listing 9: Comentar rango de líneas 5-10

```
:3,5s/^/#
```

Listing 10: Comentar todas las líneas que tengan la palabra print

```
:g/print/s/^/#
```

## 5.8. Virtual Environment

- La reutilización de código fuente (paquetes, librerías, plugins, etc.) de terceros nos permite construir software más complejo, sobre todo con menos tiempo.
- En NodeJS se usaban paquetes instalados en el directorio de trabajo y no de manera global, registrando estos paquetes en sus versiones en el archivo package.json.
- Por eso este modo de trabajo nos permite tener distintos proyectos con distintas bibliotecas, de distintas versiones, en la misma máquina, sin que existan conflictos.
- Para compartir el proyecto se debe compartir el archivo package.json y luego llamar a "npm install" para instalar las bibliotecas adecuadas para el proyecto.
- Java usa ant y maven, junto con archivos xml para realizar estas tareas.
- Python tiene **virtualenv**, para crear este espacio de trabajo.
- Python utiliza el manejador de paquetes pip.

## 5.9. Pip

- Instalemos pip, una herramienta que instalará y administrará los paquetes de programación que queramos usar en nuestros proyectos de desarrollo.

Listing 11: Instalación de pip

```
$ sudo apt-get install -y python3-pip
```

## 5.10. Garantizando posibilidades para trabajar con entornos virtuales

- Paquetes y herramientas de desarrollo más para instalar para garantizar que tengamos una configuración sólida para nuestro entorno de programación.

Listing 12: Instalaciones previas para entorno virtual

```
$ sudo apt-get install build-essential libssl-dev libffi-dev python3-dev
```

## 5.11. Instalación de paquetes para crear entornos virtuales

- Los entornos virtuales permiten tener un espacio aislado en los proyectos Python.
- Garantizando que cada proyecto pueda tener su propio conjunto de dependencias que no interrumpirán a otros proyectos.

- Manejando diferentes versiones de los paquetes. Esto es especialmente importante cuando se trabaja con paquetes de terceros.
- Puede varios entornos de programación.
- Cada entorno es un directorio en la que se ubicarán sus scripts.
- Usaremos el módulo venv, que es parte de la biblioteca estándar de Python.
- Instalemos venv escribiendo:

Listing 13: Instalación del entorno virtual

```
$ sudo apt-get install -y python3-venv
$ sudo apt-get install python3-virtualenv
```

## 5.12. Crear un directorio para entorno virtual

- Para crear un ambiente elija en qué directorio se va a crear el entorno virtual.

Listing 14: Creando directorio para entorno virtual

```
$ mkdir -p $HOME/rescobedoq/pw2-lab-c-23a/lab04/my_env
```

## 5.13. Crear entorno virtual en un directorio

- En este directorio crear un entorno virtual ejecutando el siguiente comando:

Listing 15: Creando entorno virtual

```
$ cd $HOME/rescobedoq/pw2-lab-c-23a/lab04/my_env
$ virtualenv -p python3 .
```

## 5.14. Estructura de un entorno virtual

- Estudie la estructura del entorno virtual.
- Dentro del directorio para el entorno virtual se debió crear un subdirectorio src/ con el siguiente contenido:

Listing 16: Creando entorno virtual

```
$ cd $HOME/rescobedoq/pw2-lab-c-23a/lab04/my_env
$ virtualenv -p python3 .
```

```
.
|--- bin
| |--- activate
| |--- activate.csh
| |--- activate.fish
| |--- activate.nu
```

```
| |--- activate.ps1
| |--- activate_this.py
| |--- deactivate.nu
| |--- pip
| |--- pip-3.9
| |--- pip3
| |--- pip3.9
| |--- python -> /usr/local/opt/python@3.9/bin/python3.9
| |--- python3 -> python
| |--- python3.9 -> python
| |--- wheel
| |--- wheel-3.9
| |--- wheel3
| |--- wheel3.9
|--- lib
| |--- python3.9
|--- pyenv.cfg
|--- text.txt
3 directories, 20 files
```

## 5.15. Activando entorno virtual

- En el directorio de trabajo active el entorno virtual ejecutando el script **activate**:

Listing 17: Activando entorno virtual

```
$ cd $HOME/rescobedoq/pw2-lab-c-23a/lab04/exercises/
$ source ../../my_env/bin/activate
(my_env) user@localhost:$
```

## 5.16. Trabajando dentro del entorno virtual

- Cree el **Hola Mundo** con su entorno virtual activado.
- Al finalizar todos sus ejercicios, no olvide de desactivar el entorno virtual.

Listing 18: Trabajando en el entorno virtual

```
(my_env) user@localhost:$ vim hello.py
(my_env) user@localhost:$ python3 hello.py
(my_env) user@localhost:$ deactivate
user@localhost:$
```

# 6. Ejercicios

- Ejercicios sobre matrices de tamaño NxN.

## 6.1. Matriz escalar

- Determine si una matriz es escalar.

Listing 19: esEscalar.py

```
1 def esEscalar(m):
2     d = m[0][0]
3     for i in range(len(m)):
4         for j in range(len(m)):
5             if i != j:
6                 if m[i][j] != 0:
7                     print(m[i][j])
8                     return False
9             elif m[i][j] != d:
10                print(m[i][j])
11                return False
12 return True
```

- Prueba el metodo **esEscalar()**.

Listing 20: testEsEscalar.py

```
1 import esEscalar as fu
2
3 def prueba(M):
4     if (fu.esEscalar(M)):
5         print("Si es escalar")
6     else:
7         print("No es escalar")
8
9 #Z = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
10 #Z = [[1, 2, 3], [4, 1, 6], [7, 8, 1]]
11 Z = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
12
13 prueba(Z)
```

## 6.2. Matriz unitaria

- Determine si una matriz es unitaria:

Listing 21: esUnitaria.py

```
1 import esEscalar as fu
2
3 def esUnitaria(m):
4     return m[0][0] == 1 and fu.esEscalar(m)
```

- Pruebe el método **esUnitaria()**.

Listing 22: testEsUnitaria.py

```
1 import esUnitaria as fu
2
3 def prueba(M):
4     if (fu.esUnitaria(M)):
5         print("Si es unitaria")
```

```
6 else:
7     print("No es unitaria")
8
9 #Z = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
10 #Z = [[1, 2, 3], [4, 1, 6], [7, 8, 1]]
11 Z = [[2, 0, 0], [0, 2, 0], [0, 0, 2]]
12 #Z = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
13
14 prueba(Z)
```

### 6.3. Enviar avances a repositorio GitHub

- Se recomienda que se hagan commits en cada avance sustancial, preprogramado por usted mismo, sin embargo, siempre haga commits para guardar sus avances.

Listing 23: Enviar avances a repositorio GitHub

```
$ git add .
$ git commit -m "Mensaje preciso para explicar avance"
$ git push -u origin main
```

- Cuando este trabajando en los laboratorios de la escuela y haya terminado de enviar sus avances, debe eliminar el directorio de trabajo.

Listing 24: Eliminar directorio de trabajo

```
$ rm -R $HOME/rescobedoq/
```

- Finalmente, siempre apague la computadora y deje todo en orden, como le gustaría haber encontrado el laboratorio. Ponga el ejemplo.



Listing 25: Apagar Ubuntu GNU/Linux

```
$ sudo systemctl poweroff
```



## 7. Tarea

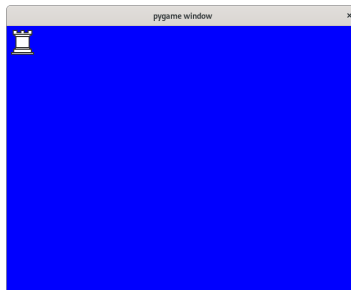
- URL GitHub de Tarea del Ajedrez <https://github.com/rescobedoq/pw2/tree/main/labs/lab04/Tarea-del-Ajedrez>.
- En esta tarea usted pondrá en práctica sus conocimientos de programación en Python para dibujar un tablero de Ajedrez.
- La parte gráfica ya está programada, usted solo tendrá que concentrarse en las estructuras de datos subyacentes.
- Con el código proporcionado usted dispondrá de varios objetos de tipo `Picture` para poder realizar su tarea:

	rock
	knight
	bishop
	queen
	king
	square

- Estos objetos estarán disponibles importando la biblioteca: **chessPictures** y estarán internamente representados con arreglos de **strings** que podrá revisar en el archivo **pieces.py**
- La clase **Picture** tiene un solo atributo: el arreglo de strings **img**, el cual contendrá la representación en caracteres de la figura que se desea dibujar.
- La clase **Picture** ya cuenta con una función implementada, no debe modificarla, pero si puede usarla para implementar sus otras funciones.
- **invColor**: recibe un color como un caracter de texto y devuelve su color negativo, también como texto, deberá revisar el archivo `colors.py` para conocer los valores negativos de cada caracter.
- La clase **Picture** contará además con varios métodos que usted deberá implementar:
  - **verticalMirror**: Devuelve el espejo vertical de la imagen.
  - **horizontalMirror**: Devuelve el espejo horizontal de la imagen.
  - **negative**: Devuelve un negativo de la imagen.
  - **join**: Devuelve una nueva figura poniendo la figura del argumento al lado derecho de la figura actual.

- **up:** Devuelve una nueva figura poniendo la figura recibida como argumento, encima de la figura actual.
  - **under:** Devuelve una nueva figura poniendo la figura recibida como argumento, sobre la figura actual.
  - **horizontalRepeat:** Devuelve una nueva figura repitiendo la figura actual al costado la cantidad de veces que indique el valor de n.
  - **verticalRepeat:** Devuelve una nueva figura repitiendo la figura actual debajo, la cantidad de veces que indique el valor de n.
- Tenga en cuenta que para implementar todos estos métodos, solo deberá trabajar sobre la representación interna de un **Picture**, es decir su atributo **img**.
  - Para dibujar una objeto **Picture** bastará importar el método **draw** de la biblioteca **interpreter** y usarlo de la siguiente manera:

```
$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from chessPictures import *
>>> from interpreter import draw
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> draw(rock)
3 directories, 20 files
```



- Ejercicios:
  - Para resolver los siguientes ejercicios solo está permitido usar ciclos, condicionales, definición de listas por comprensión, sublistas, map, join, (+), lambda, zip, append, pop, range.
  - Implemente los métodos de la clase Picture. Se recomienda que implemente la clase picture por etapas, probando realizar los dibujos que se muestran en la siguiente preguntas.
  - Usando únicamente los métodos de los objetos de la clase Picture dibuje las siguientes figuras (invoque a draw).


## 7.1. Ejercicio 1

- Código de ejercicio 1

Listing 26: Ejercicio2a.py

```
1 from pieces import *
2 from picture import *
3 from interpreter import draw
4 from colors import *
5
6 # Crear una instancia de la clase Picture con el caballo blanco
7 horse_white = Picture(KNIGHT)
8
9
10 horse_black = horse_white.changeColor('.', '@')
11 horse_combined = horse_black.join(horse_white)
12
13 horse_white2 = Picture(KNIGHT)
14 horse_black2 = horse_white.changeColor('.', '@')
15 horse_combined2 = horse_white2.join(horse_black2)
16
17 horse_block=horse_combined.under(horse_combined2)
18
19 draw(horse_block)
```

- Captura de la ejecución de ejercicio 1

 pygame window




## 7.2. Ejercicio 2

- Código de ejercicio 2

Listing 27: Ejercicio2b.py

```
1 from pieces import *
2 from picture import *
3 from interpreter import draw
4 from colors import *
5
6 # Crear una instancia de la clase Picture con el caballo blanco
7 horse_white = Picture(KNIGHT)
8
9
10 horse_black = horse_white.changeColor('.', '@')
11 horse_combined = horse_black.join(horse_white)
12
13 horse_white2 = Picture(KNIGHT)
14 horse_black2 = horse_white.changeColor('.', '@')
15 horse_combined2 = horse_black2.join(horse_white2)
16 horse_combined2 = horse_combined2.horizontalMirror()
17 horse_block = horse_combined.under(horse_combined2)
18
19 draw(horse_block)
```

- Captura de la ejecución de ejercicio 2

 pygame window




### 7.3. Ejercicio 3

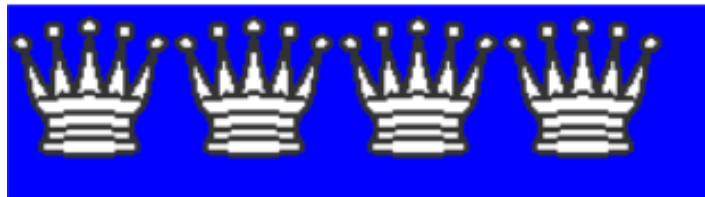
- Código de ejercicio 3

Listing 28: Ejercicio2c.py

```
1 from pieces import *
2 from picture import *
3 from interpreter import draw
4 from colors import *
5
6 # Crear una instancia de la clase Picture con el caballo blanco
7 queen_white = Picture(QUEEN)
8 queen_block = Picture(QUEEN)
9
10 for i in range(3):
11     queen_block=queen_block.join(queen_white)
12
13 draw(queen_block)
```

- Captura de la ejecución de ejercicio 3

 pygame window



## 7.4. Ejercicio 4

- Código de ejercicio 4

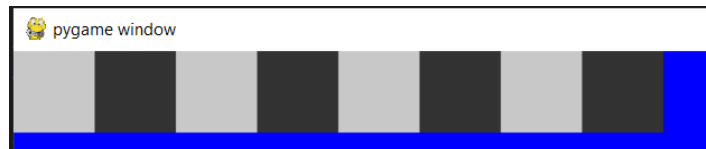
Listing 29: Ejercicio2d.py

```

1 from interpreter import draw
2 from chessPictures import *
3
4 square_white = Picture(SQUARE)
5 square_black = square_white.changeColor('_', '#')
6
7 square_block = square_white.join(square_black)
8
9 for i in range(6):
10     square_block = square_block.join(square_white if i % 2 == 0 else square_black)
11
12 draw(square_block)

```

- Captura de la ejecución de ejercicio 4



## 7.5. Ejercicio 5

- Código de ejercicio 5

Listing 30: Ejercicio2e.py

```

1 from interpreter import draw
2 from chessPictures import *
3
4 square_white = Picture(SQUARE)
5 square_black = square_white.changeColor('_', '#')
6
7 square_block = square_black.join(square_white)
8
9 for i in range(6):
10     square_block = square_block.join(square_black if i % 2 == 0 else square_white)
11
12 draw(square_block)

```

- Captura de la ejecución de ejercicio 5



## 7.6. Ejercicio 6

- Código de ejercicio 6

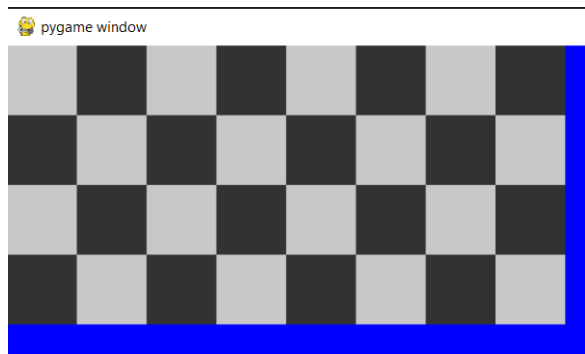
Listing 31: Ejercicio2f.py

```

1 from interpreter import draw
2 from chessPictures import *
3
4
5 square_white = Picture(SQUARE)
6 square_black = square_white.changeColor('_', '#')
7
8 # Crea una fila con colores alternados, comenzando con un cuadro blanco
9 row_pattern_white = square_white.join(square_black)
10 row_pattern_black = square_black.join(square_white)
11 row_white = row_pattern_white
12 row_black = row_pattern_black
13 for _ in range(1, 4): # Repite el patron de la fila tres veces mas
14     row_white = row_white.join(row_pattern_white)
15     row_black = row_black.join(row_pattern_black)
16
17 # Alterna las filas para crear el patron del tablero
18 board_pattern = row_white
19 for i in range(3): # Repite el patron de filas tres veces mas
20     board_pattern = board_pattern.under(row_black if i % 2 == 0 else row_white)
21
22
23 # Crea el tablero completo repitiendo el patron de filas
24 board = board_pattern
25
26 draw(board)

```

- Captura de la ejecución de ejercicio 6



## 7.7. Ejercicio 7

- Código de ejercicio 7

Listing 32: Ejercicio2g.py

```
1 from interpreter import draw
2 from chessPictures import *
3 from picture import *
4
5 # Definir las imagenes de las piezas y los cuadros
6 piezas = [rock, knight, bishop, king, queen, bishop, knight, rock]
7 pawn = Picture(PAWN)
8 square = Picture(SQUARE)
9
10 # Construir la primera fila del tablero con una roca y cuadros intercalados
11 row = rock.up(square)
12
13 for i in range(1, 8):
14     sq = square
15     if i % 2 != 0:
16         sq = square.negative()
17
18     img = piezas[i].up(sq)
19     row = row.join(img)
20
21 row_negative = row.negative()
22
23 # Construir la fila de peones negros
24 row_P = pawn.up(square.negative())
25
26 for i in range(1, 8):
27     sq = square
28     if i % 2 == 0:
29         sq = square.negative()
30
31     img = pawn.up(sq)
32     row_P = row_P.join(img)
33
34 row_P_negative = row_P.negative()
35
36 # Construir una fila vacia de cuadros
37 row_Empty = square
38
39 for i in range(1, 8):
40     sq = square
41     if i % 2 != 0:
42         sq = square.negative()
43
44     row_Empty = row_Empty.join(sq)
45
46 row_Empty_negative = row_Empty.negative()
47
48 # Construir una fila con bloques alternados de cuadros vacios y negativos
49 row_Block = row_Empty
50
51 for i in range(1, 4):
```

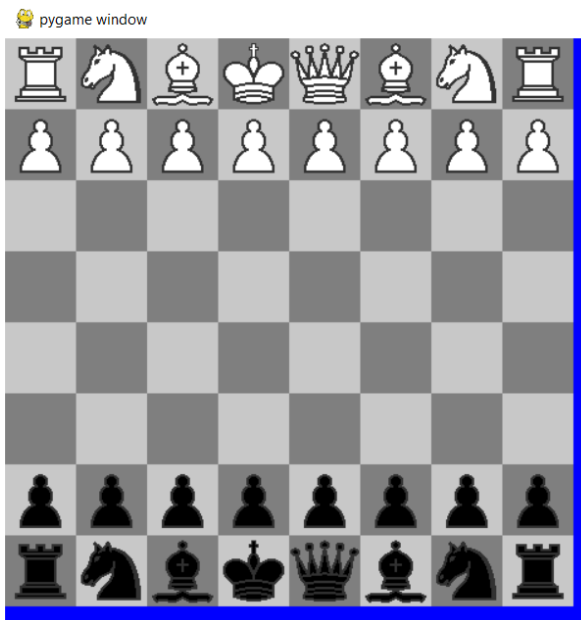


```

52     if i % 2 != 0:
53         row_Block = row_Block.under(row_Empty_negative)
54     else:
55         row_Block = row_Block.under(row_Empty)
56
57 # Construir el tablero completo
58 board = row
59 board = board.under(row_P)
60 board = board.under(row_Block)
61 board = board.under(row_P_negative)
62 board = board.under(row_negative)
63
64 # Mostrar el tablero
65 draw(board)

```

- Captura de la ejecución de ejercicio 7



## 8. Cuestionario

### ■ ¿Qué son los archivos \*.pyc?

Los archivos ".pyc" son específicos de la versión de Python que se utiliza para compilarlos. Si cambias la versión de Python, los archivos ".pyc" existentes pueden no ser compatibles y se volverán a compilar automáticamente cuando ejecutes el archivo ".py" correspondiente. Además, los archivos ".pyc" no son necesarios para ejecutar el código Python, ya que Python puede ejecutar el código fuente directamente si no se encuentra un archivo ".pyc" válido. Sin embargo, los archivos ".pyc" se generan automáticamente

### ■ para mejorar el rendimiento al ejecutar el código Python.

### ¿Para qué sirve el directorio pycache?

El directorio "pycache" está diseñado para ser gestionado automáticamente por el intérprete de Python. No se recomienda modificar ni eliminar manualmente los archivos en este directorio, ya que podría afectar la capacidad del intérprete de encontrar y utilizar los archivos de bytecode compilados adecuados. Si necesitas limpiar los archivos de bytecode compilados, puedes utilizar la opción "-B" o "-O" al invocar el intérprete de Python para evitar que se generen archivos \*.pyc o eliminar manualmente el directorio "pycache" en su totalidad.

### ¿Cuáles son los usos y lo que representa el subguión en Python?

El subguión en Python tiene varios usos y representaciones dependiendo del contexto:

- Como nombre de variable temporal: Se utiliza cuando no se necesita utilizar el valor de una variable y se desea ignorar.
- Como nombre de variable "descartable": A veces se utiliza el subguión como nombre de variable cuando el valor no es relevante y solo se necesita ejecutar una función o método.
- Como nombre de traducción o convención: En algunos casos, se utiliza el subguión como convención para indicar que una variable o método no se utiliza o no es relevante en ese contexto.

## 9. Rúbricas

### 9.1. Entregable Informe

Tabla 1: Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

## 9.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
<b>Total</b>		20		12	

## 10. Referencias

- <https://www.w3schools.com/java/default.asp>
- <https://www.geeksforgeeks.org/insertion-sort/>