
# UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ingeniería
Programa de Ingeniería Mecatrónica

---

EXAMEN N°1

"Programa ejecutable de apoyo para tratamiento de datos en el rubro crediticio"

---

## DESARROLLO DE GUIA DE EXAMEN

PROGRAMACIÓN II

**ESTUDIANTE** : **Zelada Tafur, Erlin Cecidio**

**DOCENTE** : **Asto Rodríguez, Emerson Máximo**

**CICLO** : **2020 II**

Trujillo, Perú
2020

# INDICE

# RESUMEN

En el siguiente trabajo se da a conocer sobre los resultados obtenidos al crear un programa ejecutable que sirva de ayuda en administraciones bancarias. El programa esta implementado en lenguaje de programación *Python* y consiste en una interface con diversas opciones con las que puede Registrar, Mostrar, Buscar, Modificar y mostrar cambios de Datos, toda la información de los clientes de un sistema crediticio. Los resultados obtenidos cumplieron con las expectativas y se pudo obtener una estructura dinámica y amigable con el usuario, siendo el proceso, de la siguiente manera; En primer lugar, se ve una clara aplicación de lo que se conoce como "*Programación Orientada a Objetos*", pues el código viene implementado a base del uso de "*clases*", Además de las clases, también se hiso uso de los "*métodos*" y "*objetos*" que nos ayudaron a entablar una estructura mas solida al momento de volver a usar las acciones guardadas en las clases. Finalmente, el script del programa desarrollado en Python se convirtió en un archivo ".*exe*" para así ejecutarlo con mayor facilidad, dicha conversión se realizo mediante la librería "*pyinstaller*" desde el terminal "*CMD*". El código se encuentra alojado en el repositorio de GitHub y la dirección URL lo puede encontrar en la sección de resultados.

# DESARROLLO DEL LABORATORIO

## 1.1. Resultados de la experiencia

Durante y hasta el fin del desarrollo del programa, se obtuvo 3 resultados que vale la pena destacar.

### a) Resultado 1

Como primer resultado; logramos estructurar un código en base a la teoría de programación orientada a objetos. Se destacan las principales características implementadas al crear el script en lenguaje Python.

(Alojado en: https://github.com/ezelada/ErlinZelada )

#### a.1 . *Importación*

```
import msvcrt
import os
listaClientes = []
```

Ilustración 1: Importando las librerías; "*msvcrt*" (Para Esperar tecla) y "*os*" (para limpiar pantalla), además de inicializar una variable.

### a.2 Clase *Clientes*

```
class Clientes(object):
    def __init__(self, _DNI, _apellidos, _nombres,  _monto,
_fecha, _estado, _strike1, _strike2, _strike3):

#Funcion que sera la que imprima los datos, segun se indique
    def imprimirDatos(self):

    def nivelactual(self):

# Función que nos ayuda a escribir el nivel de puntualidad al
pagar
    def nivelPuntualidad(self, _strike1, _strike2, _strike3):

# Función que nos ayuda a escribir el estado del credito
    def editarEstado(self, _estado):

# Estas funciones nos serviran para mostrar el historial
    def contarAccion(self, _strike1, _strike2, _strike3):

    def contarAccion2(self, _estado):

    def entregaHistorial(self):
```

Ilustración 2: Se muestra la clase *Clientes* con sus respectivas funciones que se encargaran de realizar las acciones repetitivas encaminadas por las funciones que se muestran en la *ilustración 3 (**El script se encuentra comprimido, Ver completo en el repositorio**)*

### a.3 *Creación de las funciones para una mejor presentación*

```
# Funcion para limpair pantalla
def LimpiarPantalla ():
# Funcion para Indicar con una tecla para continuar
def seguir ():

#Funcion del encabezado
def EncabezadoTabla ():
# Funcion de registro del cliente
```

Ilustración 3: En esta sección se encuentran las funciones que

### a.4 *Creación de la clase que reparte y mantiene el conteo de las cartas*

```
def registrarCliente():

#Función para mostrar la lista de clientes

def listadoClientes():

# Función para Buscar a un cliente
def buscarCliente():

# Función para modificar la puntualidad de pago de un
cliente
def modificarPuntualidad():

#Funcion para modificar el estado del credito, es
decir si ya esta cancelado o aun activo
def modificarEstado():

#Funcion para ver todos los ultimos cambios realizados

def consultarHistorial():
#Función para salir del menu principal
def salir():
```

Ilustración 4: En esta sección se encuentran las funciones que realizaran harán la tarea que se indique ne el menú principal y a su vez utilicen los objetos de la clase *Clientes.*

### a.5 Función principal

```
# Función main Para el menu principal
def main():

#Hacemos el llamado a la función principal
if __name__ == '__main__':
    main()
```

Ilustración 5: En esta sección se encuentra la funcion principal y el llamado de la misma

### b) Resultado 2

Como segundo resultado se obtuvo una ejecución en donde se observa en el menú principal con las 6 opciones administrativas y una séptima que es para salir del programa.

```
 _____________________________________________________
|                                                     |
|       Créditos EZETA Le da la Bienvenida            |
|_____________________________________________________|

   Menú Principal
Indique el N° de operación que desea realizar:
1.- Registrar un Cliente
2.- Ver lista de Clientes
3.- Buscar un Cliente
4.- Modificar Nivel de puntualidad
5.- Modificar estado del crédito
6.- Consultar Historial crediticio
7.- Salir
```

Ilustración 6: Menú principal del programa

### a) Resultado 3

Finalmente, el programa se dispuso a convertir el script en un archivo ejecutable, para lo que se utilizó la librería *pyinstaller*.[1] para utilizar de manera mas rápida.[2]

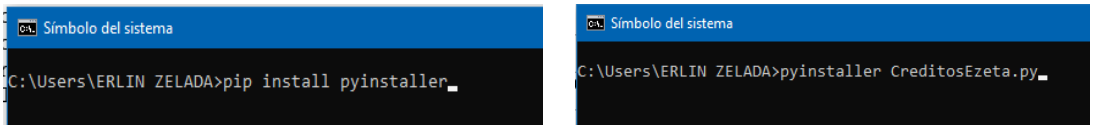


Ilustración 7: Para utilizar dicha librería se instala con el comando *pip install pyinstaller*, mientras que para convetrir al script en ejecutable se usa el comando *py installer [Archivo].py*

[1] La librería ***pyinstaller*** generar un .EXE en Windows, un .DMG en MAC o el ejecutable que utilice el sistema operativo.
[2] Dentro del ejecutable se incluye el propio intérprete de Python, y por esa razón podremos utilizarlo en cualquier ordenador sin necesidad de instalar Python previamente.

## 1.2. Recomendaciones para la repetición del trabajo.

a) Tener una cuenta configurada en GitHub (*GitHub: Where the world builds software*, 2020).

b) Tener instalado Python (*Download Python*, 2020).

c) Tener conocimientos básicos en lenguaje Python (*Contenido de la documentación de Python — documentación de Python - 3.9.1*, 2020).

d) Tener un IDE compatible con Python (Ejemplo: Visual Studio Code) (Microsoft, 2016)

e) Verificar que tengamos la librería "pyinstaller" instalada

## 1.3. Recomendaciones para ejecución de programa

Puesto que también se presenta en archivo ejecutable, puede ponerlo en marcha en cualquier computadora con sistema operativo Windows 10 x64

## 1.4. Conclusiones

a) A partir del trabajo anteriormente ejecutado se concluye que el uso de clases en el desarrollo de scripts con repeticiones y consultas constantes, ahorra el trabajo o lo hace más fácil, por no decir que sin las clases, sería imposible' lograrlo.

b) El código ha sido terminado con éxito y cumple con lo esperado, cumple con poder registrar nuevos clientes, con ver una lista de clientes registrados, buscar un cliente dentro de la lista, modificar datos de los clientes y hasta muestra un historial de los últimos cambios realizados.

c) El archivo ejecutable nos ayuda a tener un acceso más rápido a un script, en este caso -Al programa-.

# REFERENCIAS BIBLIOGRÁFICAS


*Contenido de la documentación de Python — documentación de Python - 3.9.1*. (2020). Python Docs. Recuperado 9 de diciembre de 2020, de https://docs.python.org/es/3/contents.html

*Download Python*. (2020). Python.org. Recuperado 9 de diciembre de 2020, de https://www.python.org/downloads/

*GitHub: Where the world builds software*. (2020). GitHub. Recuperado 7 de diciembre de 2020, de https://github.com/

Microsoft. (2016, 14 abril). *Visual Studio Code - Code Editing. Redefined*. https://code.visualstudio.com/