Instituto Tecnológico de Costa Rica

Campus Tecnológico Central Cartago

Escuela de Ingeniería en Computación

Taller de Programación (IC-1803)

Prof. Ing. William Mata Rodríguez

Programa #3

Parqueo

Estudiante: Dominic José Casares Aguirre c.2022085016

dcasares@estudiantec.cr

I Semestre – 22-6-2022

Abstract

The objective of this document is to serve as a documentation of the third program assigned by the teacher, which consisted of the creation of an interface in Python which seems like a parking lot program.

Keywords: *Python, program, parking lot, interface.*

Contenido

Enunciado del Proyecto

El enunciado de este trabajo empieza con la definición del proyecto como tal, el cual consiste en el desarrollo de un programa con una interfaz de tipo GUI en el lenguaje de programación Python, dicho programa simula una funcionalidad similar a la de un parqueo con sus propios registros, el cuál tendrá una configuración y sus debidas entradas y salidas de los carros, además a esto un cajero y opciones para consultar las diferentes cantidades de saldo, entradas y salidas de dinero.

Para el desarrollo de este trabajo se implementó:

- Estructuras de datos secuenciales nativas de Python (Listas, Tuplas, Strings).
- Técnicas de Iteración (Estructura For).
- Técnicas para recorridos de secuencias.
- Ciclos de Iteración While para repetición de procesos.
- Condicionales y comparadores.
- Uso del Módulo Tkinter de Python para manejar las interfaces.
- Uso del Módulo Messagebox para desplegar avisos, confirmaciones y errores.
- Uso del Módulo Pickle para manejo óptimo de archivos binarios.
- Uso del Módulo Requests para verificar la existencia de un correo electrónico.
- Uso del Módulo Win32 para apoyo del envío del correo electrónico.

Objetivos del Proyecto

Principalmente se busca la aplicación de buenas técnicas de programación básicas: documentación interna y externa del programa, reutilización de software, nombres significativos. Además, se busca:

- Validar correctamente los datos de entrada según las restricciones que se indican en cada uno de ellos.
- Usar archivos de datos.
- Aplicar y reforzar aspectos del lenguaje Python 3.
- Uso de diversos componentes del lenguaje.
- Desarrollo de funciones.

- Manejo de la técnica de iteración para repetición de procesos.
- Utilizar estructuras de datos nativas de Python tales como las secuencias (listas, tuplas, strings), diccionarios

Por otro lado, aplicar el ciclo de la metodología general para el desarrollo de programas a situaciones de mayor alcance:

- Entender el problema.
- Diseñar el algoritmo.
- Codificar el algoritmo.
- Probar y evaluar el programa.

Por último, usar algún software de control de versiones de software, y fomentar habilidades investigativas (lectura-escritura, procesos cognitivos, trabajo colaborativo, socialización del conocimiento, obtención de información, análisis, motivación, conciencia del autoaprendizaje).

Temas Investigados

- Software de control de versiones
    - ¿Qué es?
        - Consiste en una herramienta (programa), utilizado para controlar todo lo referente a los cambios en el tiempo de un archivo. Estos métodos fueron creados con el objetivo de mantener una organización y un registro exacto acerca de los cambios realizados en un programa o código, para así crear distintas “versiones” de un mismo trabajo. [1]
    - Importancia en Ingeniería de Software.
        - Según [1] en pocos casos un archivo de código o un documento de texto está terminado con la primera escritura, estos necesitan cambios y reescrituras constantes para realizar correcciones y solucionar errores. El usar un software de control de versiones es un método ideal para conservar estos cambios de manera cronológica lo que permite que sea útil para el futuro desarrollo del proyecto, además a esto, usualmente estas herramientas van

de la mano con un registro en la nube, de manera que si se llegan a ocurrir pérdidas de archivos se pueden recuperar fácilmente accediendo a la misma nube.

- Funciones utilizadas del software de control de versiones usado en el proyecto.
  - Para el desarrollo de este programa se realizó uso de una herramienta llamada "GitKraken" la cuál fue usada junto a "GitHub" para registrar los distintos avances realizados al código del archivo parqueo.py, avances tales como la finalización de una sección significativa del proyecto como la finalización de una función o la implementación de un algoritmo.

- Envío de correos electrónicos.

Mediante el uso del módulo "win32" se realizó la creación del algoritmo mostrado en la figura 1, su funcionalidad se basa en el uso de la aplicación Outlook nativa de la computadora, esto se realizó con el propósito de que los correos enviados mediante el uso del programa sean mediante el correo estudiantil.

C: > Users > Izhepix > Desktop > emailsender.py > ...

```
1  import win32com.client as win32
2  outlook = win32.Dispatch('outlook.application')
3  mail = outlook.CreateItem(0)
4  mail.To = 'emaildestino@email.ocm'
5  mail.Subject = 'Asunto'
6  mail.Body = 'Mensaje'
7
8  mail.Send()
```

Figura 1. Código del proceso para enviar un correo electrónico.

- Nuevas características de Tkinter exploradas en este proyecto.

El uso del método "register" fue esencial para la validación de entradas dentro la codificación del programa, las funcionalidades de este método fueron implementadas de tal manera que se pueda realizar un proceso por cada tecla que el usuario tocaba a la hora de escribir en una entrada de datos, posteriormente fue utilizado para desarrollar partes adicionales en la configuración.

Conclusiones del Trabajo

I. Problemas Encontrados y soluciones:

1. Actualizar datos conforme se escribe en una entry.

   Solución: Mediante el uso del método “register” y el parámetro “validatecommand”, se realiza un proceso en la función “cargarcajero” donde primeramente se determina si los datos que se están brindando son números, posteriormente a eso se actualiza el texto de las labels necesarias para reflejar los cálculos.

2. Labels no se actualizan en una función exterior.

   Solución: Dejar de pasar las labels a actualizar por parámetros y utilizarlas de forma global.

3. Crear PDF de la factura.

   Solución: Mediante el uso del módulo reportlab, el cuál se utiliza posteriormente para crear un canvas, desplegado en un archivo pdf con la información básica de la factura.

4. Algoritmo para enviar correos electrónicos.

   Solución: Debido a políticas de seguridad de Google, se imposibilitó el enviar un correo electrónico mediante sus servidores, por lo que se investigaron distintas alternativas para llegar al proceso documentado en la sección ***Temas Investigados.***

5. Crear el diccionario vacío para empezar el parqueo.

   Solución: Mediante unas validaciones en las primeras líneas se verifica si se puede leer el archivo “parqueo.dat”, si el proceso es irrealizable se crea el propio archivo con un diccionario vacío.

II. Aprendizajes Obtenidos:

- **A Nivel Educativo:**

  - Se reforzaron conocimientos del uso de estructuras de datos en Python.
  - Refuerzo en el desarrollo de GUI (Graphical User Interface) en el desarrollo de software.
  - Desarrollo de GUI con el módulo Tkinter y aprendizaje de nuevas técnicas.
  - Manipulación de datos mediante el uso de archivos de datos y el módulo Pickle.

- **A Nivel Personal:**

II. Se reforzó una visión de lo que consiste la entrega de un programa completo, en donde cada función tiene un propósito en común con las demás, aplicando conocimientos con experiencia previa.

III. Una mejor organización de tiempo para dedicarle la cantidad necesaria al desarrollo del programa.

Estadística de Tiempos

Actividad Realizada	**Horas**
Análisis del problema	8 horas
Diseño de algoritmos	10 horas
Investigación	7 horas
Programación	30 horas
Documentación interna	4 horas
Pruebas	8 horas
Elaboración del manual de usuario	1 hora, 30 minutos
Elaboración de documentación del proyecto	5 horas
Etc.	2 horas
TOTAL	75 horas y 30 minutos

Lista de Revisión del Proyecto

Concepto	Puntos originales	Avance 100/%/0	Puntos obtenidos	Análisis de resultados
Menú principal	2	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Configuración	10	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Cargar cajero	8	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Saldo del cajero	4	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Ingresos: Reales Estimados	8 8	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Entrada de vehículo	5	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Cajero del parqueo paso 1	5	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Cajero del parqueo paso 2: Registrar pago Devolver el pago Enviar correos Recibo del pago	5 3 5 4	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Cajero del parqueo paso 3	5	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Salida de vehículo: Eliminar elemento de parqueo Agregar elemento a detalle de uso Entrada automática por exceder tiempo de salida	3 3 3	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Validación de datos	8	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Ayuda (manual de usuario desplegado en el programa)	5	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
Archivos: Leer archivos Grabar archivos	3 3	100%		Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
TOTAL	**100**	100%		
Funciones desarrolladas adicionalmente				

Partes Adicionales Desarrolladas

1. Configuración.
   Se implementaron registros para verificar que los caracteres digitados en las entradas que deben ser números enteros sean únicamente números y no letras ni otros caracteres que no sirven.

```
#registros para validar entradas
verif = window_configuracion.register(solonumeros)#función para verificar que solo se ponen números
ent_espaciosparqueo.config(validate="key",validatecommand=(verif,"%P"))
ent_minsparasalir.config(validate="key",validatecommand=(verif,"%P"))
ent_moneda1.config(validate="key",validatecommand=(verif,"%P"))
ent_moneda2.config(validate="key",validatecommand=(verif,"%P"))
ent_billete1.config(validate="key",validatecommand=(verif,"%P"))
ent_billete2.config(validate="key",validatecommand=(verif,"%P"))
```

Figura 2. Codificación de las validaciones en las entradas.

Referencias Bibliográficas

[1] Borrell, G. (2006, 13 abril). *El control de versiones*. cversiones. Recuperado 15 de junio de 2022, de https://www.scribbr.es/detector-de-plagio/generador-apa/new/webpage/