UNIVERSIDAD CATÓLICA DE TEMUCO
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
FACULTAD DE INGENIERÍA

Evaluación 2 Programación I

Generación de Contratos

Nombres: Gustavo Valenzuela, Javiera Riffo, Juan Pablo Valdebenito

INFO 1120-Programación I-S1

Profesor Ignacio Lincolao

Desarrollo

Análisis del Problema:

Usted es un Ingeniero Informático recién egresado en busca de trabajo, gracias a la página InfoJobs encuentra una vacante como informático en la empresa “Corporación de Innovación y Desarrollo Tecnológico” ubicada en la ciudad de Temuco, el puesto se basa en el desarrollo de un programa sobre la creación automatizada de contratos para todos los empleados que se incorporan próximamente a la empresa.
Al convencerle la idea, envía su curriculum vitae a la empresa esperando que esta le acepte su solicitud.
Al cabo de unos días, revisa su correo donde se encuentra con un mensaje de la empresa que le indica que ha sido contratado para formar parte del proyecto "Evaluación de competencias específicas y metodologías de aprendizaje artificial 2020, ID 67703-20-JJ90.".
Una vez dentro de la empresa se te da acceso a todo lo necesario para empezar con la resolución del problema, una de las herramientas a la que se te da acceso es una base de datos que contiene dos tablas, una con la información de los futuros empleados y la segunda con la información de los salarios que le corresponderá a cada rol.

Explicación del Código:

En esta parte del Informe se explicarán los siguientes códigos:

-word_gen.py: Estructura del Word.
-data.py: Extrae información para la plantilla del contrato.
-proyecto1.py: Conexión de la base de datos con los archivos Word.
--Gtorta.py: Gráfico generado con Matplotlib.
-Gsueldo.py: Gráfico generado con Matplotlib.
-Gnacionalidad.py: Gráfico generado con Matplotlib.

word_gen.py:

Explicación Resumida:

Este código genera una función principal y única llamada **example_contract** que le da la estructura al Word que se debe generar con la ayuda de la librería docx, esta función recibe varios parámetros que corresponden a cada dato que irá en el contrato, sobre la generación del archivo Word la función tiene variables para asignar el orden, estilo y formato del documento.

Explicación General:

1) Se empieza importando **Document** de la librería **python docx** que se usa para el manejo de archivos Word y **docx.shared** que sirve para determinar tamaños y márgenes en cm y mm.

```
from docx import Document
from docx.shared import  Pt,Cm,Mm
```

2) Se define la función **example_contract**, donde se le asignan parámetros que corresponden a los datos que llevará el contrato generado.

```
def example_contract(date:str, rol:str, address:str,
```

3) Crea una variable document y le asigna **document()** para crear un archivo Word, luego le da la fuente **Book Antiqua** con la que debe aparecer el texto, en la variable **sections** divide el documento en secciones para posteriormente darle diferentes configuraciones.

```
document = Document()
font = document.styles['Normal'].font
font.name = 'Book Antiqua'
sections = document.sections
```

4) Recorre todas las secciones con un bucle for y les asigna valores a los tamaños de la hoja, márgenes y la distancia entre encabezados y pie de páginas.

```
for section in sections:
    section.page_height = Cm(35.56)
    section.page_width = Cm(21.59)
    section.top_margin = Cm(0.5)
    section.bottom_margin = Cm(1.27)
    section.left_margin = Cm(1.27)
    section.right_margin = Cm(1.27)
    section.header_distance = Mm(12.7)
    section.footer_distance = Mm(12.7)
```

5) Aquí añade la imagen que irá en el encabezado de la hoja y más abajo añade la imagen del logo de la empresa, ambas se añaden a través de la función **add_run**.

```
header = document.sections[0].header
paragraph = header.paragraphs[0]
run = paragraph.add_run()
run.add_picture("imagenes/header.png")
document.add_picture('imagenes/logo.png')
```

6) Empieza con una variable h a la cual le agrega un párrafo vacío con la función **add_paragraph**, luego llama a la variable h y junto a **add_run** se le agrega el Título, con **.bold = True** hace que el título aparezca en negrita.
Vuelve a llamar a la variable h y ahora junto a **alignment** con el valor de 1 hace que el título quede centrado, luego en **font.size** le da el tamaño al título y por último agrega un texto de solamente puntos que sirven meramente por estética y como separador.

```
h = document.add_paragraph('')
h.add_run('CONTRATO DE PRESTACIÓN DE SERVICIOS A HONORARIOS\n').bold = True
h.alignment = 1
font.size = Pt(10)
h.add_run(
```

7) Agrega un texto y le da el tamaño.

```
p = document.add_paragraph(
font.size = Pt(8)
```

8) Se crea una variable p1 y se le asigna un párrafo vacío, luego se le agregan diversos textos que corresponden a la información que lleva el contrato, deja todos los textos en negrita y algunos con variables concatenadas.

```
p1 = document.add_paragraph('')
p1.add_run('PRIMERO        :').bold = True
p1.add_run('En el marco del acuerdo de servicios prof
p1.add_run('\n SEGUNDO        :').bold = True
p1.add_run('El rol a desempeñar es de '+rol+'.')
p1.add_run('\n TERCERO        :').bold = True
p1.add_run('El plazo para la realización de la presta
p1.add_run('\n CUARTO        :').bold = True
p1.add_run('Por el servicio profesional efectivamente
```

9) Se crea una variable **table**, se le añade una tabla con 2 filas y 2 columnas y la centra con **alignment**, luego accede a la primera fila de la tabla y en esa celda guarda el texto ‘Rol’, hace lo mismo con la segunda celda de la fila, pero a esta le agrega el texto ‘Monto Bruto’, se cambia a la segunda fila, en la primera celda agrega el rol de la persona a la que se hará el contrato y lo mismo con el sueldo bruto (estos datos se sacan de la bd).

```
table = document.add_table(rows=2, cols=2)
table.alignment = 1
hdr_cells0 = table.rows[0].cells
hdr_cells0[0].text='Rol'
hdr_cells0[1].text='Monto Bruto'
hdr_cells = table.rows[1].cells
hdr_cells[0].text = rol
hdr_cells[1].text= salary
```

10) Se hace lo mismo anteriormente explicado con la diferencia que aquí en la 2 fila agrega puntos que simulan una línea donde deben ir las firmas tanto del empleador como el empleado.

```
table = document.add_table(rows=2, cols=2)
table.alignment = 1
hdr_cells0 = table.rows[0].cells[1].add_paragraph()
r = hdr_cells0.add_run()
r.add_picture('imagenes/firma.png')
hdr_cells = table.rows[1].cells
hdr_cells[0].text = '------------------------------
hdr_cells[1].text = '------------------------------
```

11) Se crea una variable **footer**, donde se le asigna el pie de página, aquí se agrega un texto y una imagen para dar término a la estructura del contrato, luego con la función **document.save** guarda el archivo con el nombre completo de la persona a la que se le creará el contrato y termina con .docx para que lo reconozca como un archivo Word.

```
footer = document.sections[0].footer
paragraph = footer.paragraphs[0]
run = paragraph.add_run('Caupolican 0455,
run.add_picture("imagenes/footer1.png")
document.save(f'{full_name}.docx')
```

Data.py:

Explicación Resumida:

Este código crea una función la cual tiene como objetivo tomar un **DataFrame** con la información de cada uno de los empleados y el índice del cual extraer los datos.

Explicación General:

1) Importamos librerías:
   Ocupamos la librería **pandas** que sirve para la manipulación de los datos.
   Ocupamos el **from** para poder llamar la función **example_contract**.

```
import pandas as pd
from word_gen import example_contract
```

2) Le asignamos el formato pd para la lectura de la DataFrame.
   el index_row lo ocupamos para designar el valor entero de la columna que se llama.

```
def singular_data_to_contract(df: pd.DataFrame, index_row:int):
```

3) Dentro de la función colocamos las variables que designaremos su utilidad.
   Junto a la df para poder extraer los datos que buscamos puesto en su lugar.
   por ejemplo, la variable **date = sub_df [‘Fecha_ingreso’]**
   Se usa para buscar la fecha de ingreso de X empleado.
   Llamamos la función **example_contract** que se encarga de llamar todas las variables que usamos dentro del def para dejarle un orden

```
def singular_data_to_contract(df: pd.DataFrame, index_row:int):
    sub_df = df.iloc[index_row]
    date = sub_df['fecha_ingreso']
    rol = sub_df['Rol']
    address = sub_df['residencia']
    rut = sub_df['rut']
    full_name = sub_df['nombre_completo']
    nationality = sub_df['nacionalidad']
    birth_date = sub_df['fecha_de_nacimiento']
    profession = sub_df['profesion']
    salary = sub_df['Sueldo']
    example_contract(date, rol, address, rut, full_name, nationality, birth_date, profession, str(salary))
```

Proyecto1.py

Explicación Resumida:

Este código genera una nueva función que permite un rango de valores de inicio y final para la creación de múltiples documentos Word.

Explicación General:

1) Importamos librerías:
   Primero instalamos e importamos la librería de sqlite3.
   Importamos pandas
   Importamos la data anterior y le creamos un filter para poder procesar datos.

```
import sqlite3
import pandas as pd
import data as filter
```

2) Hacemos la conexión y la consulta con sqlite3:
   Creamos la variable llamada **conexión**, lo que hace es que conecta la base de datos a nuestro código para así manejar de mejor manera.
   En la variable de dataframe lo que hacemos es hacer la consulta correspondiente para resolver el problema.
   Una vez teniendo la base de datos cerramos la consulta con **.close**

```
conexion = sqlite3.connect("./plantilla_word/db_personas.db")
dataframe = pd.read_sql_query("SELECT * FROM personas INNER JOIN Salarios ON personas.id_rol = Salarios.id_salarios", conexion)
conexion.close()
```

3) Creamos la función principal:
   Creamos una función para evaluar los números ingresados por el usuario. Usamos un try para poder validar si el usuario ingresa un número, en caso de no ser un número o exceda alguna de las cantidad que se le indican por pantalla el programa repita la pregunta generada hasta que sean introducidos los valores correctos.

```
def validar_entrada_numero():
    while True:
        try:
            n1 = int(input("Ingrese el primer numero entre 0 y 48: "))
            n2 = int(input("Ingrese el segundo numero entre 0 y 48: "))
            if 0 <= n1 <= 48 and 0 <= n2 <= 48:
                if n1<=n2:
                    print("Los numeros estan bien ingresados")
                    return n1, n2
                else:
                    print("n2 no puede ser mayor a n1")
            else:
                print("Los número debe estar entre 0 y 48.")
        except ValueError:
            print("El valor ingresado no corresponde a un número.")

n1,n2 = validar_entrada_numero()
```

4) Creamos un ciclo **FOR**:
   Se crea un bucle for que recorre desde el valor de n1 hasta el valor de n2 inclusive.
   Dentro del ciclo llamamos a la función filter.singular_data_to_contract, donde le pasamos el dataframe y el índice actual del bucle.
   Lo que logramos es que genere la cantidad de contratos que ingresa el usuario, por ejemplo, si desea el contrato del 0 al 12 generará un total de 13 contratos.

```
for indice in range (n1,n2+1):
    filter.singular_data_to_contract(dataframe,indice)
```

Gráfico de torta - Distribución de profesiones:

Explicación General:

1) Se comienza importando matplotlib.pyplot, Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones, y pyplot es un módulo Matplotlib que propone varias funciones sencillas para añadir elementos tales como líneas, imágenes o textos a los ejes de un gráfico.

```
import matplotlib.pyplot as plt
```

2) Se crea una lista que almacena los nombres de las diferentes profesiones (profesiones = [..]) y una lista que almacena los porcentajes de distribución correspondientes a cada profesión. (El primer porcentaje representa la primera profesión, el segundo a la segunda profesión y así sucesivamente).

```
profesiones = ["Administrador de bases de datos", "Desarrollador de Software", "Ingeniero
de sistemas", "Científico de datos", "Arquitecto de Software", "Desarrollador web", "Analista
de datos", "Ingeniero de redes", "Especialista en Seguridad informática" ]
dist_prof = [6.1 ,6.1 , 10.2, 10.2, 14.3 , 14.3 , 14.3, 20.4, 4.1]
```

3) Se crea un gráfico de tipo circular el cuál representará los datos asignados. dist_prof entrega los valores correspondientes a las “porciones” del gráfico, labels=profesiones asignará los nombres de cada sector del gráfico correspondiente a la profesión que corresponda y autopct= ‘%1.1f%%’ indica el formato que deben seguir los porcentajes presentes en el gráfico (al menos 1 cifra entera, un punto, al menos 1 cifra decimal y el símbolo %).

```
plt.pie(dist_prof, labels=profesiones, autopct= '%1.1f%%')
```

4) Se le asigna el título “Distribución de profesiones” al gráfico.

```
plt.title( ' Distribución de profesiones ')
```

5) Por último, se muestra por pantalla el gráfico generado.

```
plt.show()
```

Gráfico de barras- Sueldo promedio por profesión:

Explicación General:

1) Se comienza importando matplotlib.pyplot, Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones, y pyplot es un módulo Matplotlib que propone varias funciones sencillas para añadir elementos tales como líneas, imágenes o textos a los ejes de un gráfico.

```
import matplotlib.pyplot as plt
```

2) Se crea una lista que almacena los nombres de las diferentes profesiones (profesiones = [..]) y una lista (sueldo_prom = [..]) que almacena el sueldo promedio (en millones (CLP)) correspondiente a cada profesión. (El primer valor representa la primera profesión, el segundo a la segunda profesión y así sucesivamente).

```
profesiones = ["Administrador de bases de datos", "Desarrollador de Software",
"Ingeniero de sistemas", "Científico de datos", "Arquitecto de Software",
"Desarrollador web", "Analista de datos", "Ingeniero de redes", "Especialista en
Seguridad informática" ]
sueldo_prom = [1.5, 1.83,1.6,2,1.28,1.78,2.071,1.7,2]
```

3) Se crea un gráfico de barras el cuál representará los datos asignados. “profesiones” entrega las etiquetas en el eje X del gráfico (nombre de las profesiones), sueldo_prom: valores en el eje Y correspondientes al sueldo promedio por profesión y color = “purple” indica que las barras del gráfico serán de color púrpura.

```
plt.bar(profesiones,sueldo_prom, color = "purple")
```

4) Permite ingresar el título del eje “X” que en este caso será “Profesiones”

```
plt.xlabel("Profesiones")
```

5) Permite ingresar el título del eje “Y” que en este caso será "Sueldo promedio (millones CLP)"

```
plt.ylabel("Sueldo promedio (millones CLP)")
```

6) Permite ingresar el título del gráfico que en este caso será "Promedio de Sueldo por Profesión"

```
plt.title("Promedio de Sueldo por Profesión")
```

7) Las etiquetas en el eje X correspondientes a cada barra tendrán una rotación de 45 grados, con los bordes alineados hacia la derecha.

```
plt.xticks(rotation=45, ha='right')
```

8) Por último, se muestra por pantalla el gráfico generado.

```
plt.show()
```

Gráfico de barras- Conteo de nacionalidades:

Explicación General:

1) Se comienza importando matplotlib.pyplot, Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones, y pyplot es un módulo Matplotlib que propone varias funciones sencillas para añadir elementos tales como líneas, imágenes o textos a los ejes de un gráfico.

```
import matplotlib.pyplot as plt
```

2) Se crea una lista que almacena los nombres de las diferentes nacionalidades de los empleados (nacionalidad =[..]) y una lista (emp_nacion = [..]) que almacena la cantidad de empleados por nacionalidad. (El primer valor representa la primera nacionalidad, el segundo a la segunda profesión y así sucesivamente).

```
nacionalidad = ["Chilena", "Peruana", "Boliviana", "Argentina" ]
emp_nacion = [14,14,11,10]
```

3) Se crea un gráfico de barras el cuál representará los datos asignados. “nacionalidad” entrega las etiquetas en el eje “X” del gráfico (nombre de las nacionalidades), emp_nacion valores en el eje “Y” y color = “blue” indica que las barras del gráfico serán de color azul.

```
plt.bar(nacionalidad,emp_nacion, color = "blue")
```

4) Permite ingresar el título del eje “X” que en este caso será “Nacionalidad”

```
plt.xlabel("Nacionalidad")
```

5) Permite ingresar el título del eje “Y” que en este caso será "Cantidad de empleados"

```
plt.ylabel("Cantidad de empleados")
```

6) Permite ingresar el título del gráfico que en este caso será "Conteo de profesionales por nacionalidad".

```
plt.title("Conteo de profesionales por nacionalidad")
```

7) Las etiquetas en el eje X correspondientes a cada barra tendrán una rotación de 45 grados, con los bordes alineados hacia la derecha.

```
plt.xticks(rotation=45, ha='right')
```

8) Por último, se muestra por pantalla el gráfico generado.

```
plt.show()
```

Imágenes de los Códigos:

Proyecto1.py

```
import sqlite3
import pandas as pd
import data as filter

conexion = sqlite3.connect("./plantilla_word/db_personas.db")
dataframe = pd.read_sql_query("SELECT * FROM personas INNER JOIN Salarios ON personas.id_rol = Salarios.id_salarios", conexion)
conexion.close()

print(dataframe)

def validar_entrada_numero():
    while True:
        try:
            n1 = int(input("Ingrese el primer numero entre 0 y 48: "))
            n2 = int(input("Ingrese el segundo numero entre 0 y 48: "))
            if 0 <= n1 <= 48 and 0 <= n2 <= 48:
                if n1<=n2:
                    print("Los numeros estan bien ingresados")
                    return n1, n2
                else:
                    print("n2 no puede ser mayor a n1")
            else:
                print("Los número debe estar entre 0 y 48.")
        except ValueError:
            print("El valor ingresado no corresponde a un número.")

n1,n2 = validar_entrada_numero()

for indice in range (n1,n2+1):
    filter.singular_data_to_contract(dataframe,indice)
```

data.py

```
import pandas as pd
from word_gen import example_contract


def singular_data_to_contract(df: pd.DataFrame, index_row:int):
    sub_df = df.iloc[index_row]
    date = sub_df['fecha_ingreso']
    rol = sub_df['Rol']
    address = sub_df['residencia']
    rut = sub_df['rut']
    full_name = sub_df['nombre_completo']
    nationality = sub_df['nacionalidad']
    birth_date = sub_df['fecha_de_nacimiento']
    profession = sub_df['profesion']
    salary = sub_df['Sueldo']
    example_contract(date, rol, address, rut, full_name, nationality, birth_date, profession, str(salary))
```

Gtorta.py

```
import matplotlib.pyplot as plt

profesiones = ["Administrador de bases de datos", "Desarrollador de Software", "Ingeniero de sistemas", "Científico de datos",

dist_prof = [6.1 ,6.1 , 10.2, 10.2, 14.3 , 14.3 , 14.3, 20.4, 4.1]
plt.pie(dist_prof, labels=profesiones, autopct= '%1.1f%%')
plt.title( ' Distribución de profesiones ')
plt.show()
```

Gsueldo.py

```
import matplotlib.pyplot as plt

profesiones = ["Administrador de bases de datos", "Desarrollador de Software", "Ingeniero de sistemas", "Científico de datos",
sueldo_prom = [1.5, 1.83,1.6,2,1.28,1.78,2.071,1.7,2]
plt.bar(profesiones,sueldo_prom, color = "purple")
plt.xlabel("Profesiones")
plt.ylabel("Sueldo promedio (millones CLP)")
plt.title("Promedio de Sueldo por Profesión")
plt.xticks(rotation=45, ha='right')
plt.show()
```

Gnacionalidad.py

```
import matplotlib as mtp
import matplotlib.pyplot as plt

nacionalidad = ["Chilena", "Peruana", "Boliviana", "Argentina" ]
emp_nacion = [14,14,11,10]
plt.bar(nacionalidad,emp_nacion, color = "blue")
plt.xlabel("Nacionalidad")
plt.ylabel("Cantidad de empleados")
plt.title("Conteo de profesionales por nacionalidad")
plt.xticks(rotation=45, ha='right')
plt.show()
```

Conclusión:

En conclusión, en este proyecto se realizó un código que generaba una conexión entre la base de datos de la empresa con la plantilla Word. Además, se crea una nueva función donde el usuario interactúa con el programa y él puede elegir qué cantidad de contratos quiere generar, esta función cuenta con una validación de datos para asegurarse que el valor que ingrese el usuario sea válido para el programa.

Para nosotros el propósito general del proyecto el cual era crear un programa para generar contratos, fue cumplido con todos los requisitos solicitados por el profesor y con algunas dificultades que logramos superar como equipo con ayuda de profesores, ayudantes y materiales en plataforma Moodle.