UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA - UTEC

Tarea 2

Fecha de inicio: *02 de Mayo, 2021*
Fecha de entrega: *22 de Mayo, 2021*

Curso: *Programación 1 (1100)* – Entrega mediante: *Gradescope*

Indicaciones generales

1. Recuerda que la tarea es **individual**. Los casos de copia/plagio serán sancionados con nota cero (0) en la asignatura.

2. (a) La fecha límite de entrega es el **domingo 22 de Mayo a las 23:59 hrs**.
   (b) Es altamente recomendable no esperar hasta la última hora.
   (c) *Gradescope* desactivará automáticamente los envíos pasada dicha hora límite.
   (d) **No se aceptarán entregas atrasadas ni entregadas por otros medios.**

3. Revisa bien lo que entregas, aunque en esta oportunidad podrás entregar ilimitadas veces la tarea, la última enviada será la evaluada.

4. Recuerda que *Gradescope* corrige automáticamente tu entrega. Dicha plataforma mostrará si has realizado correctamente las pruebas y mostrará algunos mensajes en color verde. Puedes ver un ejemplo de este caso en el anexo 1.

5. Es posible que hayas subido tu entrega pero hayas modificado algo que no se debió en el template. En ese escenario, Gradescope te mostrará algunos mensajes de error. Puedes ver un ejemplo de esto en el anexo 2.

Gradescope

1. Nosotros les proporcionaremos un código base de donde deberán partir para completar dicho ejercicio. Este archivo es llamado `solution.py` y lo encontrarán en la indicación de la tarea en CANVAS.

2. Al finalizar, **solo** subir el archivo `solution.py` (NO cambiar el nombre del archivo y NO comprimirlo).

3. Cada pregunta tiene diversos casos de prueba. Para obtener la nota completa en una pregunta, el algoritmo debe obtener la respuesta correcta en dichos casos de prueba.

4. Si un caso de prueba falla, visualizarán un mensaje de error con sugerencias. **Lee el error**, revisa el código e inténtalo de nuevo.

5. Los input de los casos de prueba son confidenciales.

Indicaciones específicas

1. En el anexo B, se puede ver la plantilla de código.

2. Ustedes deben escribir dentro de la sección y a la misma altura de donde esta escrito *"Código comienza aquí"*. Además, no deben modificar nada debajo de *"Código acaba aquí"*. Recuerden tener cuidado con las indentaciones.

3. Los input del ejercicio se encuentran en la plantilla. Recuerden usar estas variables para resolver el ejercicio.

Pregunta 1: Lectura y organización de directorio telefónico - (3 pts)

Manuel ha encontrado su lista de contactos y necesita tu ayuda para organizarla. La lista contiene la siguiente información de cada persona:

Nombre Apellido Edad Teléfono

Manuel contiene una sola cadena que almacena todos sus contactos, esta posee la siguiente estructura:

'Mauricio Roizman 21 999999099 Mariana Gomez 19 998878112 Christian Ferrero 22 902991783 Manuela Sanchez 60 991725440 Daniel Gonzales 42 912006826'

Siendo *Mauricio, Mariana, Christian, Manuela y Daniel* los nombres de los cinco contactos en esta lista. Él te ha pedido organizar sus contactos en cuatro listas distintas:

- nombres = ['Mauricio', 'Mariana', 'Christian', 'Manuela', 'Daniel']
- apellidos = ['Roizman', 'Gomez', 'Ferrero', 'Sanchez', 'Gonzales']
- edades = [21, 19, 22, 60, 42]
- telefonos = [999999099, 998878112, 902991783, 991725440, 912006826]

Los datos de cada persona coinciden en el índice que poseen en cada lista. Es decir, la información de Christian se encuentra en el índice 2 en *nombres, apellidos, edades y telefonos*.

Observaciones.

- Recuerda considerar el tipo de dato de cada variable.
  - Nombre: string
  - Apellido: string
  - Edad: número entero
  - Teléfono: número entero
- La cadena de entrada es variable, no necesariamente los ejemplos mostrados.

Ejemplo 1.

Parámetros de la función :

```
directorio = 'Mayra Diaz 20 999999999'
```

Valor de retorno de la función :

```
nombres = ['Mayra']
apellidos = ['Diaz']
edades = [20]
telefonos = [999999999]
```

Ejemplo 2.

Parámetros de la función :

```
directorio = 'Maor Roizman 21 999999099 Mariana Gomez 19 998878112'
```

Valor de retorno de la función :

```
nombres = ['Maor', 'Mariana']
apellidos = ['Roizman', 'Gomez']
edades = [21, 19]
telefonos = [999999099, 998878112]
```

Ejemplo 3.

Parámetros de la función :

```
directorio = 'Ricardo Martinez 33 910823916 Miranda Vasquez 15
 928167302 David Perez 17 937618273'
```

Valor de retorno de la función :

```
nombres = ['Ricardo', 'Miranda', 'David']
apellidos = ['Martinez', 'Vasquez', 'Perez']
edades = [33, 15, 17]
telefonos = [910823916, 928167302, 937618273]
```

Indicaciones para las siguientes preguntas

En cada una de las siguientes preguntas encontrará un bloque titulado "Listas". Este hace referencia a las listas que se van a utilizar para que desarrollen sus soluciones. Estas son las listas que crearon previamente en la **pregunta 1**.

Importante: Al inicio de las siguientes preguntas (en el archivo **solution.py**) encontraran el siguiente fragmento de código:

```
nombres = self.nombres
apellidos = self.apellidos
edades = self.edades
telefonos = self.telefonos
```

Este fragmento de código permite que utilicen las listas desarrolladas en la pregunta 1. **No es necesario pasarle un nuevo parámetro a las funciones**, su solución de la pregunta 1 se encargará de llenar dichas listas, solo utilícelas.

Pregunta 2: Contactos por edad - (2 pts)

Ahora que cuentan con la información organizada en las cuatro listas, Manuel te ha pedido que cuentes la cantidad de personas que sean mayores a una determinada edad. Para realizar esto implementarás una función que reciba como parámetro dicha cantidad y retorne lo solicitado por Manuel.

Ejemplo 1.

Listas :

```
nombres = ['Mayra']
apellidos = ['Diaz']
edades = [20]
telefonos = [999999999]
```

Parámetros de la función :

```
edad_minima = 20
```

Valor de retorno de la función :

```
0
```

Ejemplo 2.

Listas :

```
nombres = ['Maor', 'Mariana']
apellidos = ['Roizman', 'Gomez']
edades = [21, 19]
telefonos = [999999099, 998878112]
```

Parámetros de la función :

```
edad_minima = 18
```

Valor de retorno de la función :

```
2
```

Ejemplo 3.

Listas :

```
nombres = ['Ricardo', 'Miranda', 'David']
apellidos = ['Martinez', 'Vasquez', 'Perez']
edades = [33, 15, 17]
telefonos = [910823916, 928167302, 937618273]
```

Parámetros de la función :

```
edad_minima = 30
```

Valor de retorno de la función :

```
1
```

Pregunta 3: Contactos por letra en el nombre - (3 pts)

Ahora Manuel desea saber todas las edades de sus contactos filtrados en base a la primera letra de su nombre. Para ello implementarás una función que reciba como parámetro dicha letra y que retorne una lista con las edades de todos los contactos que cumplan con el requisito.

Ejemplo 1.

Listas :

```
nombres = ['Mayra']
apellidos = ['Diaz']
edades = [20]
telefonos = [999999999]
```

Parámetros de la función :

```
primera_letra = 'P'
```

Valor de retorno de la función :

```
[]
```

Ejemplo 2.

Listas :

```
nombres = ['Maor', 'Mariana']
apellidos = ['Roizman', 'Gomez']
edades = [21, 19]
telefonos = [999999099, 998878112]
```

Parámetros de la función :

```
primera_letra = 'M'
```

Valor de retorno de la función :

```
[21, 19]
```

Ejemplo 3.

Listas :

```
nombres = ['Ricardo', 'Miranda', 'David']
apellidos = ['Martinez', 'Vasquez', 'Perez']
edades = [33, 15, 17]
telefonos = [910823916, 928167302, 937618273]
```

Parámetros de la función :

```
primera_letra = 'D'
```

Valor de retorno de la función :

```
[17]
```

Pregunta 4: Contactos por número telefónico - (5 pts)

Manuel necesita contactar urgentemente a uno de sus compañeros pero no se acuerda su nombre y apellido completo. Sin embargo, está seguro que su número de celular empieza en 9 y termina en 0. Para ayudarlo a reducir y alivianar la búsqueda, vas a implementar una función que retorne una lista de strings. Donde cada elemento contiene el nombre y apellido de todos los contactos que puedan ser el compañero buscado de Manuel.

Ejemplo 1.

Listas :

```
nombres = ['Mayra']
apellidos = ['Diaz']
edades = [20]
telefonos = [999999999]
```

Valor de retorno de la función :

```
[]
```

Ejemplo 2.

Listas :

```
nombres = ['Maor', 'Mariana']
apellidos = ['Roizman', 'Gomez']
edades = [21, 19]
telefonos = [999999091, 998878110]
```

Valor de retorno de la función :

```
['Mariana Gomez']
```

Ejemplo 3.

Listas :

```
nombres = ['Ricardo', 'Miranda', 'David']
apellidos = ['Martinez', 'Vasquez', 'Perez']
edades = [33, 15, 17]
telefonos = [910823916, 928167300, 937618270]
```

Valor de retorno de la función :

```
['Miranda Vasquez', 'David Perez']
```

Pregunta 5: Contactos por letra en el apellido - (7 pts)

Manuel te ha solicitado un último favor, desea obtener la información completa de todos sus contactos que tengan en su apellido al menos dos veces la misma letra. Esta letra puede variar, por lo cual será un parámetro de la función que implementarás. Además, esta debe retornar una lista donde cada elemento es un string que contiene toda la información de los contactos que cumplen el requisito.

Observación: La letra puede ser tanto mayúscula como minúscula. En ambos casos debe de considerarse.

Ejemplo 1.

Listas :

```
nombres = ['Mayra']
apellidos = ['Diaz']
edades = [20]
telefonos = [999999999]
```

Parámetros de la función :

```
letra = 'a'
```

Valor de retorno de la función :

```
[]
```

Ejemplo 2.

Listas :

```
nombres = ['Mauricio', 'Mariana']
apellidos = ['Gonzalez', 'Gozmez']
edades = [21, 19]
telefonos = [999999099, 998878112]
```

Parámetros de la función :

```
letra = 'z'
```

Valor de retorno de la función :

```
['Mauricio Gonzalez 21 999999099',  'Mariana Gozmez 19 998878112']
```

Ejemplo 3.

Listas :

```
nombres = ['Ricardo', 'Miranda', 'David']
apellidos = ['Garroguerrica', 'Rurunagar', 'Sandemetrio']
edades = [33, 15, 17]
telefonos = [910823916, 928167302, 937618273]
```

Parámetros de la función :

```
letra = 'r'
```

Valor de retorno de la función :

```
['Ricardo Garroguerrica 33 910823916', 'Miranda Rurunagar 15
 928167302']
```

Apéndices

A. Ejemplos Gradescope

Figure 1: Casos de prueba correctos en Gradescope.

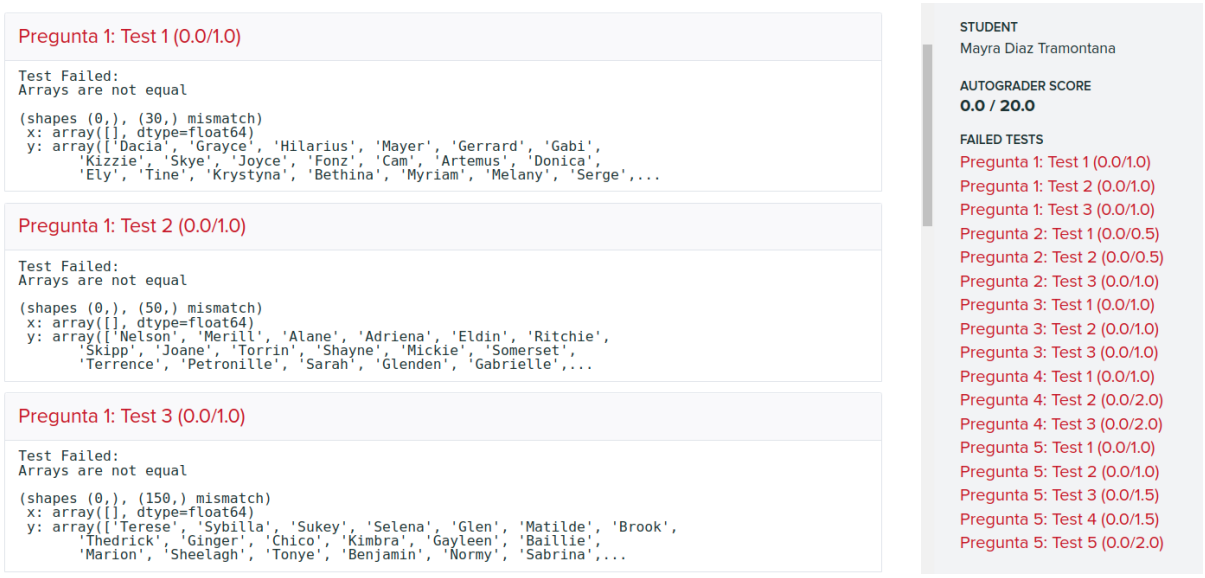

Figure 2: Entrega incorrecta en Gradescope.

B. Template

```
class Solution():
    # ============Pregunta 1 (3pts)============
    def leer_directorio(self, directorio):
        nombres = []
        apellidos = []
        edades = []
        telefonos = []

        # Codigo para Pregunta 1 comienza aqui


        # Codigo para Pregunta 1 acaba aqui

        self.nombres = nombres
        self.apellidos = apellidos
        self.edades = edades
        self.telefonos = telefonos
        return nombres, apellidos, edades, telefonos


# ============Pregunta 2 (2pts)============
    def consulta_edades(self, edad_minima):
        nombres = self.nombres
        apellidos = self.apellidos
        edades = self.edades
        telefonos = self.telefonos

        cantidad = 0
        # Codigo para Pregunta 2 comienza aqui


        # Codigo para Pregunta 2 acaba aqui
        return cantidad


    # ============Pregunta 3 (3pts)============
    # 4 puntos
    def consulta_letra_nombre(self, primera_letra):
        nombres = self.nombres
        apellidos = self.apellidos
        edades = self.edades
        telefonos = self.telefonos

        edades_letra = []
        # Codigo para Pregunta 3 comienza aqui
```

```
46
47
48          # Codigo para Pregunta 3 acaba aqui
49          return edades_letra
50
51
52      # ============Pregunta 4 (5pts)============
53      # 4 puntos
54      def consulta_numero_telefonico(self):
55          nombres = self.nombres
56          apellidos = self.apellidos
57          edades = self.edades
58          telefonos = self.telefonos
59
60          contactos = []
61          # Codigo para Pregunta 4 comienza aqui
62
63
64          # Codigo para Pregunta 4 acaba aqui
65          return contactos
66
67
68      # ============Pregunta 5 (7pts)============
69      # 6 puntos
70      def consulta_letra_apellido(self, letra):
71          nombres = self.nombres
72          apellidos = self.apellidos
73          edades = self.edades
74          telefonos = self.telefonos
75
76          contactos = []
77          # Codigo para Pregunta 5 comienza aqui
78
79
80          # Codigo para Pregunta 5 acaba aqui
81          return contactos
82
83
84  # NO MODIFICAR NADA BAJO ESTA LINEA
85  if __name__ == '__main__':
86    s = Solution()
87    text = 'Mauricio Roizman 21 999999099 Mariana Gomez 19 998878112
       Christian Ferrero 22 902991783 Manuela Sanchez 60 991725440 Daniel
       Gonzales 42 912006826'
88    print("-------- Pregunta 1 --------")
89    nombres, apellidos, edades, telefonos = s.leer_directorio(text)  # <-
       aqui puede verificar que su lectura sea correcta.
```

```
90    print(f'Nombres:\n {nombres}')
91    print(f'Apellidos:\n {apellidos}')
92    print(f'Edades:\n {edades}')
93    print(f'Telefonos:\n {telefonos}')
94
95    print("\n\n-------- Pregunta 2 --------")
96    print(s.consulta_edades(20))
97
98    print("\n\n-------- Pregunta 3 --------")
99    print(s.consulta_letra_nombre('M'))
100
101   print("\n\n-------- Pregunta 4 --------")
102   print(s.consulta_numero_telefonico())
103
104   print("\n\n-------- Pregunta 5 --------")
105   print(s.consulta_letra_apellido('r'))
106 # NO MODIFICAR NADA SOBRE ESTA LINEA
```

Listing 1: Template `solution.py`.

Se puede observar en la plantilla que en las preguntas 2, 3, 4 y 5 se encuentra al inicio el siguiente fragmento de código:

```
    nombres = self.nombres
    apellidos = self.apellidos
    edades = self.edades
    telefonos = self.telefonos

```

Este es importante puesto que extrae las listas previamente leídas en la *leer_directorio*. Por lo tanto, podrán utilizar con normalidad dichas listas en su función.