la_gran_prueba_de_sabor_(tp_prog_i).py

```
"""La Gran Prueba de Sabor (TP Prog I)

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1sEtfOrpw9RMCwhdUYHg77UDhw3J0JnV-

# **Trabajo Práctico: La Gran Prueba de Sabor**

### **Integrantes del grupo:**

1. AGUILERA, Joaquin
2. FRANK, Maximiliano
3. ROGGI, Ignacio
4. WINTER, Federico


---

# **Presentación**

**Una consultora** ha sido contratada para realizar un **análisis de mercado** y **proporcionar recomendaciones estratégicas** para la apertura de una cafetería en EE.UU.

Se le solicitó llevar a cabo un estudio utilizando los datos de la encuesta [**"La Gran Prueba de Sabor"**](https://raw.githubusercontent.com/rfordatascience/tidytuesday/refs/heads/main/data/2024/2024-05-14/coffee_survey.csv), con el fin de extraer información sobre las preferencias de los consumidores y ayudar al grupo inversor a diseñar una estrategia informada para su incursión en el sector. Esta encuesta la realizó el famoso barista *James Hoffmann* en octubre de 2023, durante una transmisión en vivo con unas *4.000 personas* en Estados Unidos.

#**Objetivo del Proyecto Final**

En este Proyecto Final, el alumno asumirá el rol de consultor y aplicará todos los conocimientos adquiridos en la materia. Deberá demostrar un uso adecuado de los elementos y construcciones del lenguaje de programación mediante la lectura, manipulación y análisis de los datos de la encuesta, así como proporcionar información útil para el inversor.

#**Consignas**

## **Etapa 0.**
**Conociendo los datos**

**1-** Descargar el archivo de la encuesta ejecutando el código que se propone a continuación.

**2-** Una vez descargado el archivos analice su contenido. ¿Qué información brinda de cada persona encuestada? ¿Todos los encuestados respondieron a todas las preguntas?
"""

#No modificar este código que le permitirá bajar el archivo que necesita para trabajar

import requests
```

```

url_coffee_survey = "https://raw.githubusercontent.com/rfordatascience/tidytuesday/refs/heads/main/data/2024/2024-05-14/coffee_survey.csv"
archivo_salida_coffee_survey = "coffee_survey.csv"

def descargarCSV(url, archivo_salida):
    print("Descargando archivo...")
    consulta = requests.get(url)
    contenido = consulta.content

    print("Guardando archivo...")
    # Abrir conexion en modo escritura
    with open(archivo_salida, "w", encoding="utf-8") as archivo:
        # Escribir el contenido de la consulta
        archivo.write(contenido.decode("utf-8"))

    print("¡Archivo descargado con éxito!")

descargarCSV(url_coffee_survey,archivo_salida_coffee_survey)

"""## **Etapa 1.**
**Analizando las respuestas**

**3-** Analice la columna *'age'* que indica el rango de edad del encuestado. ¿Qué cantidad de los encuestados que brindaron esta información pertenecen a los rangos *'<18 years old'*, *'18-24 years old'*, *'25-34 years old'*, *'35-44 years old'*, *'45-54 years old'*, *'55-64 years old'* y *'>65 years old'*, respectivamente? Para responder a esta pregunta implemente una función **contar_rangos_edad** que reciba el nombre del archivo de datos de la encuesta y devuelva un diccionario que le ayude a contar la cantidad de respuestas de cada rango etario.
"""

import csv

def contar_rangos_edad(archivo: str) -> dict:
    """Recibe la ruta de un archivo csv. Cuenta las veces que se repiten las
    respuestas en la columna 'age' y retorna un diccionario con los
    resultados."""

    with open(archivo, "r") as resultados_encuesta:

        lector = csv.DictReader(resultados_encuesta)

        dicc = dict()

        for encuestado in lector:
            if encuestado["age"] in dicc:
                dicc[encuestado["age"]] += 1
            else:
                dicc[encuestado["age"]] = 1

        return dicc


print("Conteo de rangos de edad: ", contar_rangos_edad("coffee_survey.csv"))

```

```
"""**4-** Analice la columna *'where_drink'* que indica dónde toman café los encuestados. ¿Qué diferencia encuentra entre esta columna de la anterior? ¿Qué cantidad de los encuestados que respondieron a esta pregunta toman el café *'On the go'*, *'At a cafe'*, *'At the office'*, *'At home'*, *'None of these'*, respectivamente?  Para responder a esta pregunta implemente una función **contar_lugares_consumo** que reciba el nombre del archivo de datos de la encuesta y devuelva un diccionario que le ayude a contar la cantidad de respuestas de cada lugar de consumo."""

import csv

def contar_lugares_consumo(archivo: str) -> dict:
    """Recibe la ruta de un archivo csv. Cuenta las veces que se repiten las
    respuestas en la columna 'where_drink' y retorna un diccionario con los
    resultados."""

    with open(archivo, "r") as datos:
        lector = csv.DictReader(datos)

        dicc = dict()

        for encuestado in lector:
            for lugar in encuestado["where_drink"].split(sep=", "):  # Se guardan
            # las respuestas en forma de lista al poder ser más de una.
                if lugar in dicc:
                    dicc[lugar] += 1
                else:
                    dicc[lugar] = 1

        return dicc

print("Conteo de lugares de consumo: ", contar_lugares_consumo("coffee_survey.csv"))

"""**5-** Analicen los códigos propuestos para responder a las consignas **3** y **4**. Son similares, ¿verdad? Proponga una función **procesamiento_columna**, que recibiendo el nombre del archivo y el nombre de la columna a analizar, sirva para resolver los dos casos anteriores. La función debe devolver un diccionario con las cantidades asociadas a cada uno de los valores posibles de las respuestas brindadas por los encuestados.

"""

def procesamiento_columna(archivo: str, nombre_columna: str) -> dict:
    """Recibe la ruta de un archivo csv y el nombre de una columna dentro del archivo.
    Cuenta las veces que se repiten las respuestas en la columna que se ingresa
    y retorna un diccionario con los resultados."""

    with open(archivo, "r") as datos:
        lector = csv.DictReader(datos)

        dicc = dict()

        for encuestado in lector:
            for respuesta in encuestado[nombre_columna].split(sep=", "): # Se crea
            # una lista para iterar en cada respuesta.
                if respuesta in dicc:
                    dicc[respuesta] += 1
                elif respuesta != "NA":  # Genera la nueva clave solo si es
```

```
                # distinta a "NA"
                    dicc[respuesta] = 1

        return dicc

print("Conteo de rangos de edad:", procesamiento_columna("coffee_survey.csv","age"))
print(f'\nConteo de lugares de consumo: {procesamiento_column-
a("coffee_survey.csv","where_drink")}')

"""



**6-** Pruebe la función anterior, analizando las columnas *'cups'* y *'brew'*. ¿Funciona?"""

# Prueba
# Complete para que la respuestas sean las que figuran aqui debajo

#Respuestas esperadas:
print(f'Conteo de tazas consumidas: {procesamiento_columna("coffee_survey.csv", "cups")}')
#== {'Less than 1': 348, '2': 1663, '1': 1277, '3': 473, 'More than 4': 67, '4': 121})
#Conteo de brew: {'Pod/capsule machine (e.g. Keurig/Nespresso)': 336, 'Bean-to-cup machine':
84, 'Coffee brewing machine (e.g. Mr. Coffee)': 663, 'Pour over': 2295, 'Espresso': 1518,
'French press': 735, 'Instant coffee': 130, 'Other': 677, 'Coffee extract (e.g. Cometeer)':
186, 'Cold brew': 525}
print(f'\nConteo de tipos de cafe: {procesamiento_columna("coffee_survey.csv", "brew")}')

"""## **Etapa 2.**
**Carga de la Información de los Consumidores**

**7-** Definir una clase **Consumidor** que tenga los siguientes

  - Atributos:

    `submission_id`: Identificador único del consumidor.

    `age`: Rango de edad (str).

    `gender`: Género (str).

    `cups`: Número de tazas que consume por día (str).

    `where_drink`: Lugares donde consume café (list[str]).

    `favorite`: Café preferido (str).

    `roast_level`: Nivel de tueste (str).

    `caffeine`: Tipo de cafeína (str).

    `education_level`: Nivel de educación (str).

    `employment_status`: Estado o situación laboral (str).

  - Métodos:
```

```
    `__init__`: Para inicializar los atributos.

    `__str__`: Para representar al consumidor de manera legible.

  Complete el siguiente código. Agregue todos los argumentos que necesite a los métodos.
"""

class Consumidor:

  def __init__(self, submission_id, age, gender, cups, where_drink, favorite,
               roast_level, caffeine, education_level, employment_status):
    self.submission_id = submission_id
    self.age = age
    self.gender = gender
    self.cups = cups
    self.where_drink = where_drink
    self.favorite = favorite
    self.roast_level = roast_level
    self.caffeine = caffeine
    self.education_level = education_level
    self.employment_status = employment_status

  def __str__(self):
    return (
        f"Consumidor(submission_id={self.submission_id}, age={self.age}, gender=
{self.gender}, "
        f"cups={self.cups}, where_drink={self.where_drink}, favorite={self.favorite}, "
        f"roast_level={self.roast_level}, caffeine={self.caffeine}, "
        f"education_level={self.education_level}, employment_status={self.employment_status}"
    )

"""**8-** Implemente una función llamada **cargar_consumidores** que reciba como argumento el
nombre del archivo de la encuesta y devuelva un diccionario donde la clave sea el
`submission_id` (ID del consumidor) y el valor sea una instancia de la clase `Consumidor`."""

import csv

def cargar_consumidores(archivo: str) -> dict[str, "Consumidor"]:
  """Recibe un archivo csv. Devuelve un diccionario en que sus claves son las id
  de los encuestados y los valores son una instancia de Consumidor con los datos
  de cada encuestado"""

  resultado = dict()

  with open(archivo, "r") as encuesta:
    lector = csv.DictReader(encuesta)

    for participante in lector:
        encuestado = Consumidor(
            participante["submission_id"],
            participante["age"],
            participante["gender"],
            participante["cups"],
            participante["where_drink"],
```

```
            participante["favorite"],
            participante["roast_level"],
            participante["caffeine"],
            participante["education_level"],
            participante["employment_status"],
        )

        resultado[encuestado.submission_id] = encuestado

  return resultado

"""**9-** Implemente una función llamada **filtrar_por_atributo_valor** que reciba un diccionario de consumidores como el creado en el punto anterior, un nombre de atributo (cualquiera de los atributos presentes en la clase Consumidor) y un valor de dicho atributo como argumentos. La función debe recorrer el diccionario y filtrar los consumidores, devolviendo otro diccionario cuyos consumidores hayan pasado el filtro aplicado."""

def filtrar_por_atributo_valor(
    cons: dict[str, "Consumidor"], atributo: str, valor: str
) -> dict[str, "Consumidor"]:
    """Recibe un diccionario de la forma [id_encuestado, Consumidor (instania)],
    un atributo de la clase Consumidor, y un valor que hace referencia a la
    respuesta en dicho atributo. Filtra en el diccionario aquellos encuestados
    que coincidan con el valor ingresado en su atributo pedido y retorna los
    resultados en forma de diccionario."""
    resultado = dict()

    for consumidor in cons.values(): # Recorre las instancias Consumidor
    # asociadas a los id de los encuestados.
        if getattr(consumidor, atributo) == valor:
            resultado[consumidor.submission_id] = consumidor

    return resultado

# ---------- SOLUCIÓN ALTERNATIVA ----------
'''
def filtrar_por_atributo_valor_alt(
    cons: dict[str, "Consumidor"], atributo: str, valores: tuple[str]
) -> dict[str, "Consumidor"]:
    """Recibe un diccionario de la forma [id_encuestado, Consumidor (instania)],
    un atributo de la clase Consumidor, y una tupla de valores (predefinidos)
    que hace referencia a la(s) respuesta en dicho atributo. Filtra en el
    diccionario aquellos encuestados que coincidan con el valor ingresado en su
    atributo pedido y retorna los resultados en forma de diccionario."""

    resultado = dict()

    for consumidor in cons.values(): # Recorre las instancias Consumidor
    # asociadas a los id de los encuestados.
        if getattr(consumidor, atributo) in valores:
            resultado[consumidor.submission_id] = consumidor

    return resultado
'''

```

```
"""**10-** Invocando a las funciones anteriores, ¿podría crear un diccionario que corresponda a los consumidores de género femenino (*Female*) cuya edad supere los 44 años?"""

carga      = cargar_consumidores("coffee_survey.csv")
genero     = filtrar_por_atributo_valor(carga, "gender", "Female")
edad_45_54 = filtrar_por_atributo_valor(genero, "age", "45-54 years old")
edad_55_64 = filtrar_por_atributo_valor(genero, "age", "55-64 years old")
edad_65    = filtrar_por_atributo_valor(genero, "age", ">65 years old")

lista = (edad_45_54,edad_55_64,edad_65)

total = dict()

for i in lista:  # Juntamos los diccionarios resultantes en uno solo
    total.update(i)

# ---------- SOLUCION ALTERNATIVA ----------

# edades_filtro: tuple[str] = ("45-54 years old",
#                              "55-64 years old",
#                              ">65 years old")

# filtro_edad = filtrar_por_atributo_valor_alt(genero, "age", edades_filtro)

"""## **Etapa 3.**
**Análisis de la Encuesta**

En esta sección, nos proponemos obtener información relevante sobre las preferencias de los consumidores, considerando diferentes criterios como el rango etario y el género.

Desarrolle una clase en Python que permita gestionar las respuestas de la encuesta sobre preferencias de café. Esta clase será capaz de almacenar, analizar y visualizar datos relacionados con las preferencias de café de distintos consumidores, agrupándolos por rangos de edad y género.

***Nota:*** En los análisis que realice, deberá considerar únicamente las respuestas proporcionadas, ignorando los valores NA.

**11-** Definir una clase **Encuesta** que tenga los siguientes

  - Atributos:

    `consumidores`: Diccionario que almacena los datos de los consumidores que respondieron a la encuesta. La clave es el submission_id (ID del consumidor) y el valor es una instancia de la clase `Consumidor`

    `cantidades_grupos_etarios`: Diccionario que contiene la cantidad de consumidores en cada grupo etario. La clave es el grupo etario y el valor es la cantidad de consumidores que respondieron a la encuesta en ese grupo.

    `cantidades_generos`: Diccionario que refleja la cantidad de consumidores de cada género. La clave es el género y el valor es la cantidad de consumidores que respondieron a la encuesta de ese género.

    `cafe_favorito_por_grupo_etario`: Diccionario que tiene como claves cada uno de los grupos etarios y como valor otro diccionario. Este último tiene como claves los cafés favoritos y
```

```
como valor la cantidad de consumidores que prefieren ese café dentro de ese grupo etario.

    `nivel_de_tueste_preferido_por_genero`: Diccionario que contiene como claves cada uno de los géneros y como valor otro diccionario. Este segundo diccionario tiene como claves los niveles de tueste preferidos y como valor la cantidad de consumidores que prefieren ese nivel de tueste para el género considerado.

    `maximo_nivel_educativo`: El nivel educativo al que pertenece la mayor parte de los consumidores que respondieron a la encuesta.

  - Métodos:

    `__init__`: Para inicializar los atributos.

    `analizar_rangos_edades`: Método que cuenta la cantidad de consumidores en cada rango etario.

    `analizar_generos`: Método que cuenta la cantidad de consumidores de cada género.

    `analizar_cafe_favorito_por_grupos_etarios`: Método que, para cada grupo etario, cuenta cuántos consumidores prefieren cada tipo de café.

    `analizar_nivel_de_tueste_por_genero`: Método que, para cada género, cuenta cuántos consumidores prefieren cada nivel de tueste.

    `calcular_maximo_nivel_educativo`: Método que calcula el nivel educativo que posee la mayor cantidad de consumidores.

    `graficar_grupos_etarios`: Método que realiza un gráfico de torta que muestra el porcentaje de consumidores pertenecientes a cada grupo etario.

    `graficar_cafe_favorito_por_grupos_etarios`: Método que realiza un gráfico de barras para cada grupo etario, mostrando cuántos consumidores prefieren cada tipo de café.

Complete el siguiente código.
"""

import matplotlib.pyplot as plt

class Encuesta:
  def __init__(self, archivo: str):
    self.consumidores = cargar_consumidores(archivo)
    self.cantidades_grupos_etarios = self.analizar_rangos_edades()
    self.cantidades_generos = self.analizar_generos()
    self.cafe_favorito_por_grupo_etario = self.analizar_cafe_favorito_por_grupos_etarios()
    self.nivel_de_tueste_preferido_por_genero = self.analizar_nivel_de_tueste_por_genero()
    self.maximo_nivel_educativo = self.calcular_maximo_nivel_educativo()


  def analizar_rangos_edades(self) -> dict[str,int]:
    """Método que cuenta la cantidad de consumidores en cada rango etario y
    devuelve un diccionario con los valores que le corresponden a cada rango etario"""
    resultado = dict()

    for consumidor in self.consumidores.values():
        encuestado_edad = getattr(consumidor, "age")
```

```
380
381         if encuestado_edad in resultado:
382             resultado[encuestado_edad] += 1
383         elif encuestado_edad != "NA":  # Genera la nueva clave solo si es
384         # distinta a "NA"
385             resultado[encuestado_edad] = 1
386
387     return resultado
388
389
390   def analizar_generos(self) -> dict[str,int]:
391     """Cuenta la cantidad de consumidores de cada género."""
392     resultado = dict()
393
394     for consumidor in self.consumidores.values():
395         encuestado_genero = getattr(consumidor, "gender")
396
397         if encuestado_genero in resultado:
398             resultado[encuestado_genero] += 1
399         elif encuestado_genero != "NA":  # Genera la nueva clave solo si es
400         # distinta a "NA"
401             resultado[encuestado_genero] = 1
402
403     return resultado
404
405   def analizar_cafe_favorito_por_grupos_etarios(self) -> dict[str,dict[str,int]]:
406     """Filtra las respuestas de de la columna edad de los encuestados y
407     guarda la cantidad de respuestas de los cafes favoritos en forma de diccionario"""
408     resultado = dict()
409
410     for consumidor in self.consumidores.values(): #Recorre los encuestados
411         edad = getattr(consumidor, 'age')
412         favorito = getattr(consumidor, 'favorite')
413
414         if edad != "NA" and favorito != "NA":
415             if edad not in resultado:
416                 resultado[edad] = {}
417
418             if favorito in resultado[edad]:
419                 resultado[edad][favorito] += 1
420             else:
421                 resultado[edad][favorito] = 1
422
423     return resultado
424
425   def analizar_nivel_de_tueste_por_genero(self) -> dict[str,dict[str,int]]:
426     """Filtra las respuestas de de la columna toast_level de los encuestados y
427     guarda la cantidad de respuestas de los niveles de tueste en forma de diccionario"""
428     resultado = dict()
429
430     for consumidor in self.consumidores.values(): # Recorre las clases Consumidor
431         genero = getattr(consumidor, 'gender')
432         tueste = getattr(consumidor, 'roast_level')
433
434         if genero != "NA" and tueste != "NA":
```

```
            if genero not in resultado:
                resultado[genero] = {}

            if tueste in resultado[genero]:
                resultado[genero][tueste] += 1
            else:
                resultado[genero][tueste] = 1

    return resultado

  def calcular_maximo_nivel_educativo(self) -> str:
    """Crea un diccinario con la cantidad de veces que se repiten cada nivel de
    educacion en las respuestas y luego compara los valores para calcular
    el nivel educativo que mas se repite (valor máximo)"""
    resultado = dict()

    for consumidor in self.consumidores.values():
        encuestado_educacion = getattr(consumidor, "education_level")

        if encuestado_educacion in resultado:
            resultado[encuestado_educacion] += 1
        elif encuestado_educacion != "NA":  # Genera la nueva clave solo si es
        # distinta a "NA"
            resultado[encuestado_educacion] = 1

    educ_maxima = "Master's degree"

    for nivel, cantidad in resultado.items():
        if cantidad > resultado[educ_maxima]:
            educ_maxima = nivel

    return f"La educacion con más cantidad de consumidores es {educ_maxima}"

  def graficar_grupos_etarios(self) -> None:
    """Crea un grafico de torta que represeta los porcentajes de la cantidad
    de cada grupo etario"""
    cant_rangos_etarios = self.analizar_rangos_edades()
    lista_edades = cant_rangos_etarios.keys()

    acumulador = 0
    lista_porcentajes = list()

    for i in cant_rangos_etarios.values():
      acumulador += i

    suma_porcentajes = 0

    for i in cant_rangos_etarios.values():
      porcentaje = (i*100 / acumulador) * 0.01
      lista_porcentajes.append(porcentaje)
      suma_porcentajes += porcentaje      #Verifica si la suma de porcentajes da 1

    plt.figure(figsize = (17,8))
    plt.title("Rangos Etarios",fontsize=25)   #titulo del gráfico y tamaño de la fuente
    plt.pie(lista_porcentajes, labels  = lista_edades, autopct="%0.1f %%")
```

```
490
491     print(lista_porcentajes, suma_porcentajes)
492
493     return None
494
495   def graficar_cafe_favorito_por_grupos_etarios(self) -> None:
496     cafe_fav_rango = self.analizar_cafe_favorito_por_grupos_etarios()
497
498     for edades, cafe_fav in cafe_fav_rango.items():
499       plt.figure(figsize = (17,3))
500       plt.xticks(rotation = 45, fontsize=12);
501       plt.yticks(fontsize=8);
502       plt.xlabel("Cafe", fontsize=16)
503       plt.ylabel('Cantidad',fontsize=16)
504       plt.title(edades ,fontsize=15)
505       plt.bar(cafe_fav.keys(), cafe_fav.values())
506
507     return None
508
509 """**12-** Cree un objeto de tipo `Encuesta` y cargue los datos del archivo
    *coffee_survey.csv*."""
510
511 ejemplo_encuesta = Encuesta("coffee_survey.csv") # Ejemplo de prueba
512
513 print(ejemplo_encuesta.graficar_cafe_favorito_por_grupos_etarios())
514 print(ejemplo_encuesta.graficar_grupos_etarios())
515
516 """**13-** **Conclusiones:**
517
518 Realice un análisis exhaustivo de los datos cargados en el objeto de tipo `Encuesta` recién
    creado. ¿Qué información relevante se puede extraer? Puede ayudarse de métodos del objeto para
    ver los gráficos o imprimir en pantalla información de este objeto. Reflexione sobre las
    conclusiones que se pueden obtener a partir de esta información.
519
520 Además, ¿qué recomendaciones ofrecería a su cliente para optimizar su cafetería? Por ejemplo,
    ¿a qué segmentos de clientes debería orientar su campaña de marketing para maximizar el
    impacto y atraer a más consumidores?
521 """
522
523
524
525 """# **Principales conclusiones:**
526
527 *   Los adultos de entre 25 y 44 años representan casi 3/4 de los encuestados(73.4 %). El cafe
    preferido de ese rango es el cafe filtrado (pour over coffee).
528 *   Para los rangos etarios mayores a 55 años prefieren tomar cafe por goteo normal (regular
    drip coffee).
529 *   Es interesante notar que el tipo de "Latte" esta posicionado entre los 3 mas tomados en
    todas las edades, siendo el favorito de los menores de 18 años.
530
531 A partir de los datos podemos definir que la campaña de marketing deberia estar orientada al
    sector 18-45 años y estar centrada en los 3 tipos mas populares de cafe: filtrado(pour over),
    por goteo (regular drip coffee) y latte.
532
533 Algunas ideas a implementar en la campaña de marketing:
534
```

```
* gran enfoque en redes sociales ofreciendo descuentos por participar en actividades, etiquetar en historias, etc.
* experiencias en el local como eventos de degustacion.
* Incorporar estrategias de fidelización como membresías.

*Complete aquí sus conclusiones*
"""
```