Ejercicio 1 (1 punto)

```
def lee_carreras(filename: str) -> list[CarreraFP]:
    with open(filename, encoding="utf-8") as f:
        carreras = []
        lector = csv.reader(f)
        next(lector)
        for fecha_hora, circuito, pais, seco, tiempo, primero_nombre, primero_escuderia,
         segundo_nombre, segundo_escuderia, tercero_nombre, tercero_escuderia in lector:
            fecha_hora = datetime.strptime(fecha_hora, "%Y-%m-%d %H:%M")
            seco = parsea_estado(seco)
            # También se admiteseco = seco == "Seco"
            tiempo = float(tiempo)
            podio = [
                Piloto(primero_nombre, primero_escuderia),
                Piloto(segundo_nombre, segundo_escuderia),
                Piloto(tercero_nombre, tercero_escuderia)
            ]
            carreras.append(CarreraFP(fecha_hora, circuito, pais, seco, tiempo, podio))
    return carreras

def parsea_estado(estado_str: str) -> str|None:
    res = None
    estado_str = estado_str.upper()
    if estado_str == "MOJADO":
        res = False
    elif estado_str == "SECO":
        res = True
    return res
```

TEST

```
def mostrar_iterable(it: Iterable) -> None:
    for elem in it:
        print("\t", elem)

def test_lee_carreras(carreras:list[CarreraFP])->None:
    print("Test lee_carreras")
    print("Total carreras:", len(carreras))
    print(f"Las dos primeras son:")
    mostrar_iterable(carreras[:2])
    print(f"Las dos últimas son:")
    mostrar_iterable(carreras[-2:])

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
    test_lee_carreras(carreras)
```

Ejercicio 2 (1,5 puntos)

```
# Versión con filtro por compresión y zip con bucles
def maximo_dias_sin_ganar(carreras: list[CarreraFP], piloto: str) -> int|None:
    filtradas = (c for c in carreras if c.podio[0].nombre == piloto)
    ord = sorted(filtradas, key=lambda c:c.fecha_hora)
    res = None
    if len(ord) >= 2:
        lista_dif = []
```

```
        for c1,c2 in zip(ord, ord[1:]):
            dias_dif = (c2.fecha_hora - c1.fecha_hora).days
            lista_dif.append(dias_dif)
        res = max(lista_dif)
    return res

# Versión con filtro por compresión y zip por compresion
def maximo_dias_sin_ganar2(carreras: list[CarreraFP], piloto: str) -> int|None:
    filtradas = (c for c in carreras if c.podio[0].nombre == piloto)
    ord = sorted(filtradas, key=lambda c:c.fecha_hora)
    res = None
    if len(ord) >= 2:
        gen_dif = ((c2.fecha_hora - c1.fecha_hora).days  \
                           for c1, c2 in zip(ord, ord[1:]))
        res = max(gen_dif)
    return res
```

TEST

```
def test_maximo_dias_sin_ganar(carreras: list[CarreraFP], piloto: str) -> None:
    print("Test maximo_dias_sin_ganar")
    dias = maximo_dias_sin_ganar(carreras, piloto)
    print(f"Máximo de días sin ganar para {piloto}: {dias}")

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
    test_maximo_dias_sin_ganar(carreras, "Marc Marquez")
    test_maximo_dias_sin_ganar(carreras, "Jorge Martin")
    test_maximo_dias_sin_ganar(carreras, "Freddie Mercury")
```

Ejercicio 3 (1,5 puntos)

```
# Solución 1:
# Crear un diccionario de agrupación cuyos valores son listas con nombres de pilotos
# Crear un segundo diccionario con máximos en los valores a partir del anterior
def piloto_mas_podios_por_circuito(carreras: list[CarreraFP]) -> dict[str, str]:
    dicc = defaultdict(list)
    for carrera in carreras:
        for piloto in carrera.podio:
            dicc[carrera.circuito].append(piloto.nombre)

    # Cálculo del segundo diccionario por compresión
    return {circuito: piloto_mas_podios(lista_nombres) \
                for circuito, lista_nombres in dicc.items()}

    # También:Cálculo del segundo diccionaro con bucles
    # for circuito in dicc:
    #     dicc[circuito]=piloto_mas_podios(dicc[circuito])
    # return dicc

def piloto_mas_podios(lista_nombres: list[str]) -> str:
    c = Counter(lista_nombres)
    return c.most_common(1)[0][0]
    # También cálculo del máximo con max
    # return max(c, key = c.get)
# Solución 2:
# Crear un diccionario de agrupación en el que los valores son listas con circuitos
```

```
# Esta función auxiliar se puede reutilizar en el ejercicio 6 si se añade un parámetro
# para filtrar por estado. Si el estado es None, entonces no hay filtro
# Crear un segundo diccionario con máximos en los valores a partir del anterior
def piloto_mas_podios_por_circuito2(carreras: list[CarreraFP]) -> dict[str, str]:
    d = carreras_por_circuito(carreras, None)
    return {circuito:piloto_mas_podios(lista_carreras) \
                for circuito, lista_carreras in d.items()}

def carreras_por_circuito(carreras:list[CarreraFP],  estado:str|None)->dict[str,
list[CarreraFP]]:
    res = defaultdict(list)
    for c in carreras:
        if estado == None or parsea_estado(estado) == c.seco:
            res[c.circuito].append(c)
    return res

def piloto_mas_podios2(carreras:list[CarreraFP])->str:
    pilotos_con_podio = (piloto.nombre for c in carreras for piloto in c.podio)
    c = Counter(pilotos_con_podio)
    return max(c, key=c.get)
```

TEST

```
def test_piloto_mas_podios_por_circuito(carreras: list[CarreraFP]) -> None:
    print("Test piloto_mas_podios_por_circuito")
    podios = piloto_mas_podios_por_circuito(carreras)
    print("Piloto con más podios por circuito:")
    for circuito, piloto in podios.items():
        print(f"{circuito}: {piloto}")

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
    test_piloto_mas_podios_por_circuito(carreras)
```

Ejercicio 4 (2 puntos)

```
def escuderias_con_solo_un_piloto(carreras: list[CarreraFP]) -> list[str]:
    d = pilotos_por_escuderia(carreras)
    return [escuderia for escuderia, pilotos in d.items() \
             if len(pilotos) == 1]

def pilotos_por_escuderia(carreras: list[CarreraFP]) -> dict[str, set[str]]:
    res = defaultdict(set)
    for c in carreras:
        for piloto in c.podio:
            res[piloto.escuderia].add(piloto.nombre)
    return res
```

TEST

```
def test_escuderias_con_solo_un_piloto(carreras: list[CarreraFP]) -> list[str]:
    print("Test escuderias_con_solo_un_piloto")
    escuderias = escuderias_con_solo_un_piloto(carreras)
    print("Escuderías con un solo piloto:", escuderias)

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
```

```
    test_escuderias_con_solo_un_piloto(carreras)
```

Ejercicio 5 (2 puntos)

```
# Solución 1: Conteo de las rachas y cálculo del máximo mezclados
def piloto_racha_mas_larga_victorias_consecutivas(carreras: list[CarreraFP], año:
int|None = None) -> tuple[str|None, int|None]:
    carreras = (carrera for carrera in carreras if año == None \
        or año == carrera.fecha_hora.year)
    carreras_ordenadas = sorted(carreras, key = lambda carrera: carrera.fecha_hora)

    max_consecutivas = 0
    piloto_max = None
    contador_actual = 0
    piloto_actual = None

    for carrera in carreras_ordenadas:
        ganador = carrera.podio[0].nombre

        if ganador == piloto_actual:
            contador_actual += 1
        else:
            piloto_actual = ganador
            contador_actual = 1
        if contador_actual >= max_consecutivas:
            max_consecutivas = contador_actual
            piloto_max = ganador

    return piloto_max, max_consecutivas

# Solución 2:
# Calcular una lista con el nombre del piloto y la longitud de las rachas con zip
# Calcular el máximo
def piloto_racha_mas_larga_victorias_consecutivas2(carreras: list[CarreraFP], año:
int|None=None) -> tuple[str|None, int|None]:

    filtradas = (c for c in carreras if año == None or c.fecha_hora.year == año)
    ordenadas = sorted(filtradas, key = lambda c:c.fecha_hora)
    rachas = []

    # Esquema de tratamiento de parejas de elementos
    len_racha = 0 # Contador para la longitud de las rachas
    for c1, c2 in zip(ordenadas, ordenadas[1:]):
        len_racha += 1
        # Cuando cambia el nombre del piloto con respecto al anterior
        # termina su racha, por lo que guardamos la longitud en la lista
        # y reiniciamos la cuenta poniendo el contador a 0
        if c1.podio[0].nombre != c2.podio[0].nombre:
            rachas.append((c1.podio[0].nombre, len_racha))
            len_racha = 0
    return max(rachas, key = lambda t:t[1])
```

TEST

```
def test_piloto_racha_mas_larga_victorias_consecutivas(carreras: list[CarreraFP], año:
int|None=None) -> None:
```

```
    print("Test piloto_racha_mas_larga_victorias_consecutivas")
    piloto, victorias = piloto_racha_mas_larga_victorias_consecutivas(carreras, año)
    print(f"Piloto con más victorias consecutivas para año {año}: {piloto} con
{victorias} victorias")

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
    test_piloto_racha_mas_larga_victorias_consecutivas(carreras, 2024)
    test_piloto_racha_mas_larga_victorias_consecutivas(carreras, None)
```

Ejercicio 6 (2 puntos)

```
# Solución 1:
def ultimos_ganadores_por_circuito(carreras: list[CarreraFP], n: int, estado: str) ->
dict[str, str]:
    d = defaultdict(list)
    for carrera in carreras:
        # También se puede parsear el estado para la condición
        # if parsea_estado(estado) == carrera.seco:
        if estado == "Seco" and carrera.seco or estado == "Mojado" and not carrera.seco:
            d[carrera.circuito].append((carrera.fecha_hora, carrera.podio[0].nombre))
    for circuito in d:
        lista = sorted(d[circuito], reverse = True)[:n]
        d[circuito] = [x[1] for x in lista]
    return d

# Solución 2:
# Reutiliza la función carreras por_circuito definida en el ejercicio 3
def ultimos_ganadores_por_circuito2(carreras: list[CarreraFP], n:int, estado: str) ->
dict[str, list[str]]:
    d = carreras_por_circuito(carreras, estado)
    # El diccionario d tiene listas de CarreraFP en los valores
    res = dict()
    for circuito, lista_carreras in d.items():
        ordenados = sorted((c for c in lista_carreras),\
                           key = lambda c: c.fecha_hora, reverse = True)[:n]
        res[circuito] = [c.podio[0].nombre for c in ordenados]
    return res
```

TEST

```
def test_ultimos_ganadores_por_circuito(carreras: list[CarreraFP], n: int, estado: str)
-> None:
    print("Test ultimos_ganadores_por_circuito")
    ganadores = ultimos_ganadores_por_circuito(carreras, n, estado)
    print(f"Últimos {n} ganadores por circuito en estado {estado}:")
    for circuito, pilotos in ganadores.items():
        print(f"'{circuito}': {pilotos}")

if __name__ == "__main__":
    carreras = lee_carreras("data/mundial_motofp.csv")
    test_ultimos_ganadores_por_circuito(carreras, 2, "Seco")
    test_ultimos_ganadores_por_circuito(carreras, 2, "Mojado")
```