**Ejercicio 1**

```
def lee_reparaciones(fichero):
    with open(fichero, encoding="utf-8") as f:
        lector = csv.reader(f, delimiter=";")
        next(lector)
        res = []
        for (numero_referencia, centro, fecha_entrada, fecha_reparacion,
                numero_serie, tipo, descripcion_problema, fecha_compra,
                precio_reparacion) in lector:
            fecha_entrada = parsea_fecha(fecha_entrada)
            fecha_reparacion = parsea_fecha(fecha_reparacion)
            fecha_compra = parsea_fecha(fecha_compra)
            precio_reparacion = float(precio_reparacion)
            res.append(Reparacion(numero_referencia,
                centro, fecha_entrada, fecha_reparacion, numero_serie,
                tipo, descripcion_problema, fecha_compra, precio_reparacion))
    return res

def parsea_fecha(str_fecha):
    return datetime.strptime(str_fecha, "%d/%m/%Y").date()
```

**Ejercicio 2**

```
def calcula_recaudacion(reparaciones, centro, tipo=None):
    return sum(r.precio_reparacion for r in reparaciones
        if r.centro == centro and not esta_en_garantia(r)
            and (tipo is None or r.tipo == tipo))

def esta_en_garantia(r):
    return (r.fecha_compra < date(2022, 1, 1)
            and r.fecha_compra + timedelta(365 * 2) > r.fecha_entrada
            or r.fecha_compra >= date(2022, 1, 1)
            and r.fecha_compra + timedelta(365 * 3) > r.fecha_entrada)
```

**Ejercicio 3**

```
def reparaciones_mas_largas(reparaciones, año, n, centro=None):
    return sorted(( (r.numero_ref, dias_reparacion(r)) for r in reparaciones
        if r.fecha_entrada.year == año
        and (centro is None or r.centro == centro)),
        key = lambda t:t[1], reverse = True)[:n]

def dias_reparacion(r):
    return (r.fecha_reparacion - r.fecha_entrada).days
```

**Ejercicio 4**

```
def centro_mas_rapido(reparaciones, centros):
    d = agrupa_reparaciones_por_centros_filtrada(reparaciones, centros)

    d_medias = {}
    for centro, lista_reparaciones in d.items():
        suma = sum(días_reparacion(r) for r in lista_reparaciones)
        d_medias[clave] = suma / len(valor)

    return min(d_medias.items(), key=lambda t:t[1])[0]

def agrupa_reparaciones_por_centros_filtrada(reparaciones, centros):
    d = dict()
    for reparacion in reparaciones:
        if reparacion.centro in centros:
            if reparacion.centro in d:
                d[reparacion.centro].append(reparacion)
            else:
                d[reparacion.centro] = [reparacion]
       return d
```

**Ejercicio 5**

```
def centros_experimentados_en(reparaciones, keywords):
    d = agrupa_centros_por_keywords(reparaciones, keywords)

    return {keyword:ordena_centros(lista_centros)
        for keyword, lista_centros in d.items()}

def agrupa_centros_por_keywords(reparaciones, keywords):
    res = dict()

    for reparacion in reparaciones:
        keywords_rep = keywords_reparacion_min(reparacion)
        for keyword in keywords:
            clave = keyword.lower()
            if clave in keywords_rep:
                if clave in res:
                    res[clave].append(reparacion.centro)
                else:
                    res[clave] = [reparacion.centro]
    return res

def keywords_reparacion_min(reparacion):
    return set(keyword.lower()
        for keyword in reparacion.descripcion_problema.split())

def ordena_centros (lista_centros):
    c = Counter(lista_centros)
    ord = sorted(c.items(), key=lambda t:t[1], reverse=True)

    return [centro for centro, _ in ord]
```

**Ejercicio 6**

```
def dias_entre_reparaciones(reparaciones):
    d = agrupa_reparaciones_por_num_serie(reparaciones)
    return {num_serie:calcula_lista_dias(lista_reparaciones)
        for num_serie, lista_reparaciones in d.items()}

def agrupa_reparaciones_por_num_serie(reparaciones):
    d = dict()
    for reparacion in reparaciones:
        if reparacion.numero_serie in d:
            d[reparacion.numero_serie].append(reparacion)
        else:
            d[reparacion.numero_serie] = [reparacion]
    return d

def dias_funcionamiento(reparacion):
    return (reparacion.fecha_entrada - reparacion.fecha_compra).days

def calcula_lista_dias(reparaciones):
    lista_ord = sorted(reparaciones, key=lambda t: t.fecha_entrada)
    res = [dias_funcionamiento(lista_ord[0])]
    for t1, t2 in zip(lista_ord, lista_ord[1:]):
        dias_entre_reparaciones = (t2.fecha_entrada - t1.fecha_reparacion).days
        res.append(dias_entre_reparaciones)
    return res


######################################################################
#  Funciones de test
######################################################################

def test_calcula_recaudacion(lista):
    print("\nTest de la función test_calcula_recaudacion")
    print(
        "\tRecaudación en Sevilla en reparación de portátiles:",
        calcula_recaudacion(lista, "Sevilla", tipo="portátil"),
    )
    print(
        "\tRecaudación en Sevilla en todo tipo de reparaciones:",
        calcula_recaudacion(lista, "Sevilla"),
    )

def test_reparaciones_mas_largas(lista):
    print("\nTest de la función test_reparaciones_mas_largas")
    print(
        "\tReparación más larga en Sevilla en 2019:",
        reparaciones_mas_largas(lista, 2019, 1, centro="Sevilla"),
    )
    print(
        "\tLas tres reparaciones más largas en 2020:",
        reparaciones_mas_largas(lista, 2020, 3),
    )
```

```python
def test_centro_mas_rapido(lista):
    print("\nTest de la función test_centro_mas_rapido")
    print(
        "\tCentro más rápido entre Sevilla, Huelva y Cádiz:",
        centro_mas_rapido(lista, {"Sevilla", "Huelva", "Cádiz"}),
    )

def test_centros_experimentados_en(lista):
    print("\nTest de la función test_centros_experimentados_en")
    print(
        "\tListado ordenado de los centros con más experiencia en pantallas y
baterías:"
    )
    print(centros_experimentados_en(lista, {"pantalla", "batería"}))

def test_dias_entre_reparaciones(lista):
    print("\nTest de la función test_dias_entre_reparaciones")
    print("\tDías transcurridos entre reparaciones de los dispositivos:")
    print(dias_entre_reparaciones(lista))


######################################################################
#  Programa principal
######################################################################
if __name__ == "__main__":
    print("\nTest de lal función lee_reparaciones")
    lista = lee_reparaciones("../datos/reparaciones.csv")
    print("\tNúmero total de reparaciones:", len(lista))
    print("\tEstos son los 3 primeros registros:")
    print(lista[:3], "\n")

    test_calcula_recaudacion(lista)
    test_reparaciones_mas_largas(lista)
    test_centro_mas_rapido(lista)
    test_centros_experimentados_en(lista)
    test_dias_entre_reparaciones(lista)
```