Task 1 - Flights

Recommended time to complete: ~40 mins
You are given a list of flight routes, where each route is represented by a list containing three elements:

1. The departure country code (str)
2. The destination country code (str)
3. The flight time in hours (int)
4. Cost of flight (int)

```
flights = [
    ['NYC', 'LON', 7, 450],   # New York City to London
    ['TOK', 'SYD', 10, 780],  # Tokyo to Sydney
    ['PAR', 'DXB', 6, 380],   # Paris to Dubai
    ['SFO', 'HKG', 14, 920],  # San Francisco to Hong Kong
    ['RIO', 'CPT', 11, 850],  # Rio de Janeiro to Cape Town
    ['AMS', 'BKK', 11, 720],  # Amsterdam to Bangkok
    ['LAX', 'AKL', 13, 890],  # Los Angeles to Auckland
    ['IST', 'SIN', 10, 680],  # Istanbul to Singapore
    ['YVR', 'DEL', 14, 950],  # Vancouver to Delhi
    ['MEX', 'BCN', 12, 820],  # Mexico City to Barcelona
    ['JNB', 'PEK', 15, 1050], # Johannesburg to Beijing
    ['ZRH', 'BUE', 14, 980],  # Zurich to Buenos Aires
    ['CAI', 'SEL', 12, 790],  # Cairo to Seoul
    ['MIA', 'MLE', 18, 1200], # Miami to Malé (Maldives)
    ['LIS', 'YYZ', 8, 520]    # Lisbon to Toronto
]
```

Copy the provided list into your Python editor.

a) Create a function called `total_flight_times(flights)` that calculates and returns the total flight time for all routes in the given list. [2m]

b) Create a function called `longest_flight(flights)` that finds and returns the route with the longest flight time. The function should return a list containing the departure country, destination country, flight time of the longest flight, and price of it. [2m]

c) In the main part of your program:

1. Call the `total_flight_time()` function and print the result.

2. Call the `longest_flight()` function and print the result in the format:
`Longest flight: {departure} to {destination}, {time} hours, ${price}"`
[1m]

d)

1. Create a function called `find_route(flights, start, end)` that finds a route between two given countries. If a direct route exists, return the flight time. If no direct route exists, return -1.
2. Prompt for a route, separate the countries by a space (i.e. `NYC LON`). Find whether this route exists or not. Validation and re-prompting is required.
- If it does, display the output as such:
`{departure} to {destination} exists, flight time is {time} hours`
- If not, display the output as such:
`{departure} to {destination} does not exist`
[4m]

e)

1. Write a Python function called `update_flight_times(flights, delay)` that takes the original list of flights and the delay time as parameters, and returns a new list with updated flight times.
2. The CrowdStrike global outage has caused significant delays in air travel. As a result, all flights have been delayed by 20 hours. Represent this information in this format:
`Updated time: {departure} to {destination}, {time} hours (originally {original_time} hours)`
[5m]

f) As a result of the fiasco, that is, CrowdStrike's outage, all flights are given a discount, in varying amounts. Note that the following metric is based on original flight times.

- Short length flights (< 8 hours) are given 5% discounts
- Medium length flights (8-11 hours) are given 10% discounts
- Long length flights (>11 hours) are given 15% discounts

1. Write a function `discount_prices(flights)` that takes the original list of flights, updates the new price, and returns it in a list, with an extra element which is the discount (i.e. 5, 10, 15).
2. Call the `discount_prices()` function and represent the output as such:
`Updated price, {discount}% discount: {departure} to {destination},`

`${price} (originally ${original_price})`

[7m]

SAVE YOUR CODE TO `FLIGHTS.py`