The following code is an example of how you can use an expert.

- Import one of the APIs
    - from bingmaps import BingMaps
- Import these libraries
    - import numpy as np
    - from datetime import datetime
    - import time
    - import flexpolyline as fp
- Insert your input (two coordinates in the [lat, lng]-format)
    - source = np.array([41.91394911573667, 12.518699879396078])
    - destination = np.array([41.923769235641124, 12.494524115637745])
- Replace KEY with your key
    - bing = BingMaps(key="KEY")
- Compute the query
    - response = bing.query(source, destination)
    - The response is a direction.py object
- Optionally, you can specify the departure
    - bing.query(source, destination, departure)
    - For each API, the departure is in ISO 8601-format except for ArcGIS (UNIX-format)
    - Note that in GoogleMaps you can only specify future dates
    - ISO 8601-format
        - departure = "2023-10-31T09:00:00+02:00"
    - UNIX-format
        - arcgis_departure = "1699599600"
    - Note that Mapbox does not accept time zones and seconds
        - valid_mapbox_departure = "2023-10-31T09:00"
        - invalid_mapbox_departure = "2023-10-31T09:00:00+02:00"
        - invalid_mapbox_departure = "2023-10-31T09:00:00"
        - Since Mapbox does not let you set time zones, I remove it from 'departure'. It is not a problem since the APIs automatically set the timezone of the source location. Hence, I use only 'valid_mapbox_departure' for Mapbox and 'departure' for the others
            - departure = "2023-10-31T09:00:00"
    - To simplify this, set date and time once
        - date_time = datetime(year=2023, month=11, day=10, hour=8, minute=0, second=0)
        - departure = date_time.isoformat()
        - mapbox_departure = str(date_time.isoformat())[:-3]
        - arcgis_departure = int(time.mktime(date_time.timetuple()))
    - Compute the query
        - response = bing.query(source, destination, departure)
        - If you want to compute the path

        - response = bing.query(source, destination, departure, True)
        - response = bing.query(source, destination, compute_path=True)
- Note that if an exception occurs, ‘response’ will be None
- Output
    - response.distance # the distance in meters
    - response.duration # the duration (considering traffic) in seconds
    - response.path # a numpy array of polylines ['BFmq6_H4htsC', 'BF-t6_HohtsC', ...]
    - Decode the polylines into an array of coordinates [[lat1, lng1], [lat2, lng2], ...] (the boolean value set to True indicate that the input is a list)
        - print(fp.decode(response.path, is_list=True))
    - Decode the single polyline into an array of [lat, lng]
        - print(fp.decode(response.path[0]))
- Note that in the output path, the first node [0] is the source, [1] is the first next node (along the suggested path) and [n] is the destination
- Note that the tolerance distance (in meters) between two consecutive nodes differs between each expert. Thus, the suggested nodes are different between each expert and their path lengths differ. Not every API has the option to modify the tolerance!