Python Client for Google Maps Services

Release 4.10.0

Jan 27, 2023

Contents

CHAPTER 1

googlemaps Module

class googlemaps.**Client**(*key=None*, *client_id=None*, *client_secret=None*, *timeout=None*, *connect_timeout=None*, *read_timeout=None*, *retry_timeout=60*, *requests_kwargs=None*, *queries_per_second=60*, *queries_per_minute=6000*, *channel=None*, *retry_over_query_limit=True*, *experience_id=None*, *requests_session=None*, *base_url='https://maps.googleapis.com'*)

Bases: object

Performs requests to the Google Maps API web services.

Parameters

- **key** (*string*) – Maps API key. Required, unless "client_id" and "client_secret" are set. Most users should use an API key.
- **client_id** (*string*) – (for Maps API for Work customers) Your client ID. Most users should use an API key instead.
- **client_secret** (*string*) – (for Maps API for Work customers) Your client secret (base64 encoded). Most users should use an API key instead.
- **channel** (*str*) – (for Maps API for Work customers) When set, a channel parameter with this value will be added to the requests. This can be used for tracking purpose. Can only be used with a Maps API client ID.
- **timeout** (*int*) – Combined connect and read timeout for HTTP requests, in seconds. Specify "None" for no timeout.
- **connect_timeout** (*int*) – Connection timeout for HTTP requests, in seconds. You should specify read_timeout in addition to this option. Note that this requires requests >= 2.4.0.
- **read_timeout** (*int*) – Read timeout for HTTP requests, in seconds. You should specify connect_timeout in addition to this option. Note that this requires requests >= 2.4.0.
- **retry_timeout** (*int*) – Timeout across multiple retriable requests, in seconds.

- **queries_per_second** (*int*) – Number of queries per second permitted. Unset queries_per_minute to None. If set smaller number will be used. If the rate limit is reached, the client will sleep for the appropriate amount of time before it runs the current query.
- **queries_per_minute** (*int*) – Number of queries per minute permitted. Unset queries_per_second to None. If set smaller number will be used. If the rate limit is reached, the client will sleep for the appropriate amount of time before it runs the current query.
- **retry_over_query_limit** (*bool*) – If True, requests that result in a response indicating the query rate limit was exceeded will be retried. Defaults to True.
- **experience_id** (*str*) – The value for the HTTP header field name 'X-Goog-Maps-Experience-ID'.
- **requests_kwargs** (*dict*) – Extra keyword arguments for the requests library, which among other things allow for proxy auth to be implemented. See the official requests docs for more info: http://docs.python-requests.org/en/latest/api/#main-interface
- **requests_session** (*requests.Session*) – Reused persistent session for flexibility.
- **base_url** (*string*) – The base URL for all requests. Defaults to the Maps API server. Should not have a trailing slash.

Raises

- **ValueError** – when either credentials are missing, incomplete or invalid.
- **NotImplementedError** – if connect_timeout and read_timeout are used with a version of requests prior to 2.4.0.

addressvalidation(*addressLines*, *regionCode=None*, *locality=None*, *enableUspsCass=None*)

The Google Maps Address Validation API returns a verification of an address See https://developers.google.com/maps/documentation/address-validation/overview request must include parameters below. :param addressLines: The address to validate :type addressLines: array :param regionCode: (optional) The country code :type regionCode: string :param locality: (optional) Restrict to a locality, ie:Mountain View :type locality: string :param enableUspsCass For the "US" and "PR" regions only, you can optionally enable the Coding Accuracy Support System (CASS) from the United States Postal Service (USPS) :type locality: boolean

clear_experience_id()

Clears the experience ID for the HTTP header field name 'X-Goog-Maps-Experience-ID' if set.

directions(*origin*, *destination*, *mode=None*, *waypoints=None*, *alternatives=False*, *avoid=None*, *language=None*, *units=None*, *region=None*, *departure_time=None*, *arrival_time=None*, *optimize_waypoints=False*, *transit_mode=None*, *transit_routing_preference=None*, *traffic_model=None*)

Get directions between an origin point and a destination point.

Parameters

- **origin** (*string, dict, list, or tuple*) – The address or latitude/longitude value from which you wish to calculate directions.
- **destination** (*string, dict, list, or tuple*) – The address or latitude/longitude value from which you wish to calculate directions. You can use a place_id as destination by putting 'place_id:' as a prefix in the passing parameter.
- **mode** (*string*) – Specifies the mode of transport to use when calculating directions. One of "driving", "walking", "bicycling" or "transit"
- **waypoints** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – Specifies an array

of waypoints. Waypoints alter a route by routing it through the specified location(s). To influence route without adding stop prefix the waypoint with *via*, similar to *waypoints = ["via:San Francisco", "via:Mountain View"]*.

- **alternatives** (*bool*) – If True, more than one route may be returned in the response.
- **avoid** (*list or string*) – Indicates that the calculated route(s) should avoid the indicated features.
- **language** (*string*) – The language in which to return results.
- **units** (*string*) – Specifies the unit system to use when displaying results. "metric" or "imperial"
- **region** (*string*) – The region code, specified as a ccTLD ("top-level domain" two-character value.
- **departure_time** (*int or datetime.datetime*) – Specifies the desired time of departure.
- **arrival_time** (*int or datetime.datetime*) – Specifies the desired time of arrival for transit directions. Note: you can't specify both departure_time and arrival_time.
- **optimize_waypoints** (*bool*) – Optimize the provided route by rearranging the waypoints in a more efficient order.
- **transit_mode** (*string or list of strings*) – Specifies one or more preferred modes of transit. This parameter may only be specified for requests where the mode is transit. Valid values are "bus", "subway", "train", "tram", "rail". "rail" is equivalent to ["train", "tram", "subway"].
- **transit_routing_preference** (*string*) – Specifies preferences for transit requests. Valid values are "less_walking" or "fewer_transfers"
- **traffic_model** – Specifies the predictive travel time model to use. Valid values are "best_guess" or "optimistic" or "pessimistic". The traffic_model parameter may only be specified for requests where the travel mode is driving, and where the request includes a departure_time.

Return type list of routes

distance_matrix(*origins*, *destinations*, *mode=None*, *language=None*, *avoid=None*, *units=None*, *departure_time=None*, *arrival_time=None*, *transit_mode=None*, *transit_routing_preference=None*, *traffic_model=None*, *region=None*)

Gets travel distance and time for a matrix of origins and destinations.

Parameters

- **origins** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – One or more addresses, Place IDs, and/or latitude/longitude values, from which to calculate distance and time. Each Place ID string must be prepended with 'place_id:'. If you pass an address as a string, the service will geocode the string and convert it to a latitude/longitude coordinate to calculate directions.
- **destinations** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – One or more addresses, Place IDs, and/or lat/lng values , to which to calculate distance and time. Each Place ID string must be prepended with 'place_id:'. If you pass an address as a string, the service will geocode the string and convert it to a latitude/longitude coordinate to calculate directions.

- **mode** (*string*) – Specifies the mode of transport to use when calculating directions. Valid values are "driving", "walking", "transit" or "bicycling".
- **language** (*string*) – The language in which to return results.
- **avoid** (*string*) – Indicates that the calculated route(s) should avoid the indicated features. Valid values are "tolls", "highways" or "ferries".
- **units** (*string*) – Specifies the unit system to use when displaying results. Valid values are "metric" or "imperial".
- **departure_time** (*int or datetime.datetime*) – Specifies the desired time of departure.
- **arrival_time** (*int or datetime.datetime*) – Specifies the desired time of arrival for transit directions. Note: you can't specify both departure_time and arrival_time.
- **transit_mode** (*string or list of strings*) – Specifies one or more preferred modes of transit. This parameter may only be specified for requests where the mode is transit. Valid values are "bus", "subway", "train", "tram", "rail". "rail" is equivalent to ["train", "tram", "subway"].
- **transit_routing_preference** (*string*) – Specifies preferences for transit requests. Valid values are "less_walking" or "fewer_transfers".
- **traffic_model** – Specifies the predictive travel time model to use. Valid values are "best_guess" or "optimistic" or "pessimistic". The traffic_model parameter may only be specified for requests where the travel mode is driving, and where the request includes a departure_time.
- **region** (*string*) – Specifies the prefered region the geocoder should search first, but it will not restrict the results to only this region. Valid values are a ccTLD code.

Return type matrix of distances. Results are returned in rows, each row containing one origin paired with each destination.

elevation(*locations*)

Provides elevation data for locations provided on the surface of the earth, including depth locations on the ocean floor (which return negative values)

Parameters **locations** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – List of latitude/longitude values from which you wish to calculate elevation data.

Return type list of elevation data responses

elevation_along_path(*path*, *samples*)

Provides elevation data sampled along a path on the surface of the earth.

Parameters

- **path** (*string, dict, list, or tuple*) – An encoded polyline string, or a list of latitude/longitude values from which you wish to calculate elevation data.
- **samples** (*int*) – The number of sample points along a path for which to return elevation data.

Return type list of elevation data responses

find_place(*input*, *input_type*, *fields=None*, *location_bias=None*, *language=None*)

A Find Place request takes a text input, and returns a place. The text input can be any kind of Places data, for example, a name, address, or phone number.

Parameters

- **input** (*string*) – The text input specifying which place to search for (for example, a name, address, or phone number).
- **input_type** (*string*) – The type of input. This can be one of either 'textquery' or 'phonenumber'.
- **fields** (*list*) – The fields specifying the types of place data to return. For full details see: https://developers.google.com/places/web-service/search#FindPlaceRequests
- **location_bias** (*string*) – Prefer results in a specified area, by specifying either a radius plus lat/lng, or two lat/lng pairs representing the points of a rectangle. See: https://developers.google.com/places/web-service/search#FindPlaceRequests
- **language** (*string*) – The language in which to return results.

Return type result dict with the following keys: status: status code candidates: list of places

geocode(*address=None*, *place_id=None*, *components=None*, *bounds=None*, *region=None*, *language=None*)

Geocoding is the process of converting addresses (like `"1600 Amphitheatre Parkway, Mountain View, CA"`) into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map.

Parameters

- **address** (*string*) – The address to geocode.
- **place_id** (*string*) – A textual identifier that uniquely identifies a place, returned from a Places search.
- **components** (*dict*) – A component filter for which you wish to obtain a geocode, for example: `{'administrative_area': 'TX','country': 'US'}`
- **bounds** (*string or dict with northeast and southwest keys.*) – The bounding box of the viewport within which to bias geocode results more prominently.
- **region** (*string*) – The region code, specified as a ccTLD ("top-level domain") two-character value.
- **language** (*string*) – The language in which to return results.

Return type list of geocoding results.

geolocate(*home_mobile_country_code=None*, *home_mobile_network_code=None*, *radio_type=None*, *carrier=None*, *consider_ip=None*, *cell_towers=None*, *wifi_access_points=None*)

The Google Maps Geolocation API returns a location and accuracy radius based on information about cell towers and WiFi nodes given.

See https://developers.google.com/maps/documentation/geolocation/intro for more info, including more detail for each parameter below.

Parameters

- **home_mobile_country_code** (*string*) – The mobile country code (MCC) for the device's home network.
- **home_mobile_network_code** (*string*) – The mobile network code (MCC) for the device's home network.
- **radio_type** (*string*) – The mobile radio type. Supported values are lte, gsm, cdma, and wcdma. While this field is optional, it should be included if a value is available, for more accurate results.
- **carrier** (*string*) – The carrier name.

- **consider_ip** (*bool*) – Specifies whether to fall back to IP geolocation if wifi and cell tower signals are not available. Note that the IP address in the request header may not be the IP of the device.
- **cell_towers** (*list of dicts*) – A list of cell tower dicts. See https://developers.google.com/maps/documentation/geolocation/intro#cell_tower_object for more detail.
- **wifi_access_points** (*list of dicts*) – A list of WiFi access point dicts. See https://developers.google.com/maps/documentation/geolocation/intro#wifi_access_point_object for more detail.

get_experience_id()
Gets the experience ID for the HTTP header field name 'X-Goog-Maps-Experience-ID'

Returns The experience ID if set

Return type str

nearest_roads(*points*)
Find the closest road segments for each point

Takes up to 100 independent coordinates, and returns the closest road segment for each point. The points passed do not need to be part of a continuous path.

Parameters **points** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – The points for which the nearest road segments are to be located.

Return type A list of snapped points.

place(*place_id*, *session_token=None*, *fields=None*, *language=None*, *reviews_no_translations=False*, *reviews_sort='most_relevant'*)
Comprehensive details for an individual place.

Parameters

- **place_id** (*string*) – A textual identifier that uniquely identifies a place, returned from a Places search.
- **session_token** (*string*) – A random string which identifies an autocomplete session for billing purposes.
- **fields** – The fields specifying the types of place data to return, separated by a comma. For full details see: https://cloud.google.com/maps-platform/user-guide/product-changes/#places
- **language** (*string*) – The language in which to return results.
- **reviews_no_translations** (*bool*) – Specify reviews_no_translations=True to disable translation of reviews; reviews_no_translations=False (default) enables translation of reviews.
- **reviews_sort** (*string*) – The sorting method to use when returning reviews. Can be set to most_relevant (default) or newest.

Return type result dict with the following keys: result: dict containing place details html_attributions: set of attributions which must be displayed

places(*query=None*, *location=None*, *radius=None*, *language=None*, *min_price=None*, *max_price=None*, *open_now=False*, *type=None*, *region=None*, *page_token=None*)
Places search.

Parameters

- **query** (*string*) – The text string on which to search, for example: "restaurant".

- **location** (*string, dict, list, or tuple*) – The latitude/longitude value for which you wish to obtain the closest, human-readable address.
- **radius** (*int*) – Distance in meters within which to bias results.
- **language** (*string*) – The language in which to return results.
- **min_price** (*int*) – Restricts results to only those places with no less than this price level. Valid values are in the range from 0 (most affordable) to 4 (most expensive).
- **max_price** (*int*) – Restricts results to only those places with no greater than this price level. Valid values are in the range from 0 (most affordable) to 4 (most expensive).
- **open_now** (*bool*) – Return only those places that are open for business at the time the query is sent.
- **type** (*string*) – Restricts the results to places matching the specified type. The full list of supported types is available here: https://developers.google.com/places/supported_types
- **region** (*string*) – The region code, optional parameter. See more @ https://developers.google.com/places/web-service/search
- **page_token** (*string*) – Token from a previous search that when provided will returns the next page of results for the same search.

Return type result dict with the following keys: results: list of places html_attributions: set of attributions which must be displayed next_page_token: token for retrieving the next page of results

places_autocomplete(*input_text*, *session_token=None*, *offset=None*, *origin=None*, *location=None*, *radius=None*, *language=None*, *types=None*, *components=None*, *strict_bounds=False*)

Returns Place predictions given a textual search string and optional geographic bounds.

Parameters

- **input_text** (*string*) – The text string on which to search.
- **session_token** (*string*) – A random string which identifies an autocomplete session for billing purposes.
- **offset** (*int*) – The position, in the input term, of the last character that the service uses to match predictions. For example, if the input is 'Google' and the offset is 3, the service will match on 'Goo'.
- **origin** (*string, dict, list, or tuple*) – The origin point from which to calculate straight-line distance to the destination (returned as distance_meters). If this value is omitted, straight-line distance will not be returned.
- **location** (*string, dict, list, or tuple*) – The latitude/longitude value for which you wish to obtain the closest, human-readable address.
- **radius** (*int*) – Distance in meters within which to bias results.
- **language** (*string*) – The language in which to return results.
- **types** (*string*) – Restricts the results to places matching the specified type. The full list of supported types is available here: https://developers.google.com/places/web-service/autocomplete#place_types
- **components** (*dict*) – A component filter for which you wish to obtain a geocode. Currently, you can use components to filter by up to 5 countries for example: `{'country': ['US', 'AU']}`

- **strict_bounds** (*bool*) – Returns only those places that are strictly within the region defined by location and radius.

Return type list of predictions

places_autocomplete_query(*input_text*, *offset=None*, *location=None*, *radius=None*, *language=None*)

Returns Place predictions given a textual search query, such as "pizza near New York", and optional geographic bounds.

Parameters

- **input_text** (*string*) – The text query on which to search.
- **offset** (*int*) – The position, in the input term, of the last character that the service uses to match predictions. For example, if the input is 'Google' and the offset is 3, the service will match on 'Goo'.
- **location** (*string, dict, list, or tuple*) – The latitude/longitude value for which you wish to obtain the closest, human-readable address.
- **radius** (*number*) – Distance in meters within which to bias results.
- **language** (*string*) – The language in which to return results.

Return type list of predictions

places_nearby(*location=None*, *radius=None*, *keyword=None*, *language=None*, *min_price=None*, *max_price=None*, *name=None*, *open_now=False*, *rank_by=None*, *type=None*, *page_token=None*)

Performs nearby search for places.

Parameters

- **location** (*string, dict, list, or tuple*) – The latitude/longitude value for which you wish to obtain the closest, human-readable address.
- **radius** (*int*) – Distance in meters within which to bias results.
- **region** (*string*) – The region code, optional parameter. See more @ https://developers.google.com/places/web-service/search
- **keyword** (*string*) – A term to be matched against all content that Google has indexed for this place.
- **language** (*string*) – The language in which to return results.
- **min_price** (*int*) – Restricts results to only those places with no less than this price level. Valid values are in the range from 0 (most affordable) to 4 (most expensive).
- **max_price** (*int*) – Restricts results to only those places with no greater than this price level. Valid values are in the range from 0 (most affordable) to 4 (most expensive).
- **name** (*string or list of strings*) – One or more terms to be matched against the names of places.
- **open_now** (*bool*) – Return only those places that are open for business at the time the query is sent.
- **rank_by** (*string*) – Specifies the order in which results are listed. Possible values are: prominence (default), distance
- **type** (*string*) – Restricts the results to places matching the specified type. The full list of supported types is available here: https://developers.google.com/places/supported_types

- **page_token** (*string*) – Token from a previous search that when provided will returns the next page of results for the same search.

Return type result dict with the following keys: status: status code results: list of places html_attributions: set of attributions which must be displayed next_page_token: token for retrieving the next page of results

places_photo(*photo_reference*, *max_width=None*, *max_height=None*)

Downloads a photo from the Places API.

Parameters

- **photo_reference** (*string*) – A string identifier that uniquely identifies a photo, as provided by either a Places search or Places detail request.
- **max_width** (*int*) – Specifies the maximum desired width, in pixels.
- **max_height** (*int*) – Specifies the maximum desired height, in pixels.

Return type iterator containing the raw image data, which typically can be used to save an image file locally. For example:

```
f = open(local_filename, 'wb')
for chunk in client.places_photo(photo_reference, max_width=100):
    if chunk:
        f.write(chunk)
f.close()
```

reverse_geocode(*latlng*, *result_type=None*, *location_type=None*, *language=None*)

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

Parameters

- **latlng** (*string, dict, list, or tuple*) – The latitude/longitude value or place_id for which you wish to obtain the closest, human-readable address.
- **result_type** (*string or list of strings*) – One or more address types to restrict results to.
- **location_type** (*list of strings*) – One or more location types to restrict results to.
- **language** (*string*) – The language in which to return results.

Return type list of reverse geocoding results.

set_experience_id(**experience_id_args*)

Sets the value for the HTTP header field name 'X-Goog-Maps-Experience-ID' to be used on subsequent API calls.

Parameters **experience_id_args** (*string varargs*) – the experience ID

snap_to_roads(*path*, *interpolate=False*)

Snaps a path to the most likely roads travelled.

Takes up to 100 GPS points collected along a route, and returns a similar set of data with the points snapped to the most likely roads the vehicle was traveling along.

Parameters

- **path** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – The path to be snapped.

- **interpolate** (*bool*) – Whether to interpolate a path to include all points forming the full road-geometry. When true, additional interpolated points will also be returned, resulting in a path that smoothly follows the geometry of the road, even around corners and through tunnels. Interpolated paths may contain more points than the original path.

Return type A list of snapped points.

snapped_speed_limits(*path*)

Returns the posted speed limit (in km/h) for given road segments.

The provided points will first be snapped to the most likely roads the vehicle was traveling along.

Parameters **path** (*a single location, or a list of locations, where a location is a string, dict, list, or tuple*) – The path of points to be snapped.

Return type dict with a list of speed limits and a list of the snapped points.

speed_limits(*place_ids*)

Returns the posted speed limit (in km/h) for given road segments.

Parameters **place_ids** (*str or list*) – The Place ID of the road segment. Place IDs are returned by the snap_to_roads function. You can pass up to 100 Place IDs.

Return type list of speed limits.

static_map(*size*, *center=None*, *zoom=None*, *scale=None*, *format=None*, *maptype=None*, *language=None*, *region=None*, *markers=None*, *path=None*, *visible=None*, *style=None*)

Downloads a map image from the Maps Static API.

See https://developers.google.com/maps/documentation/maps-static/intro for more info, including more detail for each parameter below.

Parameters

- **size** – Defines the rectangular dimensions of the map image.
- **center** (*dict or list or string*) – Defines the center of the map, equidistant from all edges of the map.
- **zoom** (*int*) – Defines the zoom level of the map, which determines the magnification level of the map.
- **scale** (*int*) – Affects the number of pixels that are returned.
- **format** (*string*) – Defines the format of the resulting image.
- **maptype** (*string*) – defines the type of map to construct. There are several possible maptype values, including roadmap, satellite, hybrid, and terrain.
- **language** (*string*) – defines the language to use for display of labels on map tiles.
- **region** (*string*) – defines the appropriate borders to display, based on geo-political sensitivities.
- **markers** (*StaticMapMarker*) – define one or more markers to attach to the image at specified locations.
- **path** (*StaticMapPath*) – defines a single path of two or more connected points to overlay on the image at specified locations.
- **visible** (*list of dict*) – specifies one or more locations that should remain visible on the map, though no markers or other indicators will be displayed.
- **style** (*list of dict*) – defines a custom style to alter the presentation of a specific feature (roads, parks, and other features) of the map.

Return type iterator containing the raw image data, which typically can be used to save an image file locally. For example:

```
f = open(local_filename, 'wb')
for chunk in client.static_map(size=(400, 400),
                               center=(52.520103, 13.404871),
                               zoom=15):
    if chunk:
        f.write(chunk)
f.close()
```

timezone(*location*, *timestamp=None*, *language=None*)

Get time zone for a location on the earth, as well as that location's time offset from UTC.

Parameters

- **location** (*string, dict, list, or tuple*) – The latitude/longitude value representing the location to look up.
- **timestamp** (*int or datetime.datetime*) – Timestamp specifies the desired time as seconds since midnight, January 1, 1970 UTC. The Time Zone API uses the timestamp to determine whether or not Daylight Savings should be applied. Times before 1970 can be expressed as negative values. Optional. Defaults to `datetime.utcnow()`.
- **language** (*string*) – The language in which to return results.

Return type dict

CHAPTER 2

googlemaps.exceptions Module

Defines exceptions that are thrown by the Google Maps client.

exception googlemaps.exceptions.**ApiError**(*status*, *message=None*)
Bases: Exception

Represents an exception returned by the remote API.

exception googlemaps.exceptions.**HTTPError**(*status_code*)
Bases: *googlemaps.exceptions.TransportError*

An unexpected HTTP error occurred.

exception googlemaps.exceptions.**Timeout**
Bases: Exception

The request timed out.

exception googlemaps.exceptions.**TransportError**(*base_exception=None*)
Bases: Exception

Something went wrong while trying to execute the request.

Python Module Index

g

Index