
fingertips_{py}
Release 0.2

May 13, 2019

Contents:

1	Introduction	1
1.1	Installation	1
1.2	Usage	1
1.3	Example	2
1.4	Licence	2
1.4.1	metadata.py	2
1.4.2	api_calls.py	5
1.4.3	retrieve_data.py	6
1.4.4	area_data.py	7
2	Indices and tables	9
	Python Module Index	11

This is a python package to interact with Public Health England's [Fingertips](<https://fingertips.phe.org.uk/>) data tool. Fingertips is a major repository of population and public health indicators for England. The site presents the information in many ways to improve accessibility for a wide range of audiences ranging from public health professionals and researchers to the general public. The information presented is a mixture of data available from other public sources, and those that are available through user access agreements with other organisations. The source of each indicator presented is available using the `get_metadata_for_indicator()` function.

This package can be used to load data from the Fingertips API into python for further use.

1.1 Installation

This packaged should be installed using pip:

```
` pip install fingertips_py `
```

Or it can be compiled from source (still requires pip):

```
` pip install git+https://github.com/PublicHealthEngland/PHDS_fingertips_py.git `
```

1.2 Usage

`fingertips_py` should be imported and used in line with standard python conventions. It is suggested that if the whole package is to be imported

then the following convention is used:

```
` import fingertips_py as ftp `
```

The package returns data in a variety of types dependent on the function.

For more information on any function, you can use:

```
` help(*fingertips_py function name*) `
```

Or you can view the documents [here](<https://fingertips-py.readthedocs.io/en/latest/>).

1.3 Example

This is an example of a workflow for retrieving data for the indicator on *Healthy Life Expectancy at Birth* from the *Public Health Outcomes Framework* profile.

```
““ import fingertips_py as ftp
```

```
phof      =      ftp.get_profile_by_name('public      health      outcomes      framework')      phof_meta      =  
ftp.get_metadata_for_profile_as_dataframe(phof['Id']) indicator_meta = phof_indicators[phof_meta['Indicator']].str.contains('Healthy')  
print(indicator_meta)
```

```
Indicator ID Indicator ...
```

```
0 90362 0.1i - Healthy life expectancy at birth ... 55 92543 2.05ii - Proportion of children aged 2-2½yrs r... ... ““
```

We can see that the *Healthy life expectancy at birth* indicator has an id of 90362. The data for this indicator at all geographical breakdowns can be retrieved using `get_data_for_indicator_at_all_available_geographies()`

```
` healthy_life_data = ftp.get_data_for_indicator_at_all_available_geographies(90362)
```

1.4 Licence

This project is released under the [GPL-3](<https://opensource.org/licenses/GPL-3.0>) licence.

1.4.1 metadata.py

Calls used to retrieve metadata about areas, ages, sexes, value notes, calculation methods, rates, and indicator metadata.

```
fingertips_py.metadata.get_age_from_id(age_id)
```

Returns an age name from given id.

Parameters `age_id` – Age id used in Fingertips as an integer

Returns Age or age range as a string

```
fingertips_py.metadata.get_age_id(age)
```

Returns an ID for a given age.

Parameters `age` – Search term of an age or age range as a string

Returns Code used in Fingertips to represent the age or age range as a string

```
fingertips_py.metadata.get_all_ages()
```

Returns a dictionary of all the age categories and their IDs.

Returns Age codes used in Fingertips in a tuple

```
fingertips_py.metadata.get_all_areas()
```

Retreives all area types.

Returns A dictionary of all area types used in Fingertips

fingertips_{py}.metadata.get_all_profiles()

Returns all profiles.

Returns A dictionary of all profiles in Fingertips including information on domains and sequencing

fingertips_{py}.metadata.get_all_sexes()

Returns a tuple of all sex categories and their IDs.

Returns Sex categories used in Fingertips with associated codes as a tuple

fingertips_{py}.metadata.get_all_value_notes()

Returns a dictionary of all value notes and their IDs.

Returns Data value notes and their associated codes that are used in Fingertips as a list of tuples

fingertips_{py}.metadata.get_area_type_ids_for_profile(profile_id)

Returns a list of area types used within a given profile.

Parameters profile_id – ID used in Fingertips to identify a profile as integer or string

Returns A list of area types used within a given profile

fingertips_{py}.metadata.get_area_types_as_dict()

Returns all area types and related information such as ID and name.

Returns A dictionary of area types

fingertips_{py}.metadata.get_area_types_for_profile(profile_id)

Retrieves all the area types that have data for a given profile.

Parameters profile_id – ID used in Fingertips to identify a profile as integer or string

Returns A list of dictionaries of area types with relevant information

fingertips_{py}.metadata.get_areas_for_area_type(area_type_id)

Returns a dictionary of areas that match an area type id given the id as integer or string.

Parameters area_type_id – ID of area type (ID of General Practice is 7 etc) used in Fingertips as integer or string

Returns A dictionary of dictionaries with area codes and the names of those areas

fingertips_{py}.metadata.get_domains_in_profile(profile_id)

Returns the domain IDs for a given profile.

Parameters profile_id – ID used in Fingertips to identify a profile as integer or string

Returns A list of domain IDs

fingertips_{py}.metadata.get_metadata(indicator_ids=None, domain_ids=None, profile_ids=None)

Returns a dataframe object of metadata for a given indicator, domain, and/or profile given the relevant IDs. At least one of these IDs has to be given otherwise an error is raised.

Parameters

- **indicator_ids** – [OPTIONAL] Number used to identify an indicator within Fingertips as integer or string
- **domain_ids** – [OPTIONAL] Number used to identify a domain within Fingertips as integer or string
- **profile_ids** – [OPTIONAL] ID used in Fingertips to identify a profile as integer or string

Returns A dataframe object with metadata for the given IDs or an error if nothing is specified

fingertips_{py}.metadata.**get_metadata_for_all_indicators** (*include_definition='no', include_system_content='no'*)

Returns the metadata for all indicators in a dataframe.

Parameters

- **include_definition** – optional to include definitions
- **include_system_content** – optional to include system content

Returns dataframe of all indicators

fingertips_{py}.metadata.**get_metadata_for_all_indicators_from_csv**()

Returns a dataframe from the csv of all metadata for all indicators.

Returns A dataframe of all metadata for all indicators

fingertips_{py}.metadata.**get_metadata_for_domain_as_dataframe** (*group_ids*)

Returns a dataframe of metadata for a given domain ID or list of domain IDs.

Parameters **group_ids** – Number or list of numbers used to identify a domain within Fingertips as integer or string

Returns Dataframe object with metadata for the indicators for a given domain ID

fingertips_{py}.metadata.**get_metadata_for_indicator** (*indicator_number*)

Returns the metadata for an indicator given the indicator number as integer or string.

Parameters **indicator_number** – Number used to identify an indicator within Fingertips as integer or string

Returns A dictionary of metadata for the given indicator

fingertips_{py}.metadata.**get_metadata_for_indicator_as_dataframe** (*indicator_ids*)

Returns a dataframe of metadata for a given indicator ID or list of indicator IDs.

Parameters **indicator_ids** – Number or list of numbers used to identify an indicator within Fingertips as integer or string

Returns Dataframe object with metadata for the indicator ID

fingertips_{py}.metadata.**get_metadata_for_profile_as_dataframe** (*profile_ids*)

Returns a dataframe of metadata for a given profile ID or list of profile IDs.

Parameters **profile_ids** – ID or list of IDs used in Fingertips to identify a profile as integer or string

Returns Dataframe object with metadata for the indicators for a given group ID

fingertips_{py}.metadata.**get_multiplier_and_calculation_for_indicator** (*indicator_number*)

Returns the multiplier and calculation method for a given indicator.

Parameters **indicator_number** – Number used to identify an indicator within Fingertips as integer or string

Returns A tuple of multiplier and calculation method from Fingertips metadata

fingertips_{py}.metadata.**get_profile_by_id** (*profile_id*)

Returns a profile as a dictionary which contains information about domains and sequencing.

Parameters **profile_id** – ID used in Fingertips to identify a profile as integer or string

Returns A dictionary of information about the profile

fingertips_{py}.metadata.**get_profile_by_name** (*profile_name*)

Returns a profile object given a name to search - try to be specific to get better results.

Parameters `profile_name` – A string or part of a string that is used as the profile name

Returns A dictionary of the profile metadata including domain information or an error message

`fingertips_py.metadata.get_sex_from_id(sex_id)`

Returns a sex name given an id.

Parameters `sex_id` – ID used in Fingertips to represent the sex as integer

Returns Sex category as string

`fingertips_py.metadata.get_sex_id(sex)`

Returns an ID for a given sex.

Parameters `sex` – Sex category as string (Case sensitive)

Returns ID used in Fingertips to represent the sex as integer

`fingertips_py.metadata.get_value_note_id(value_note)`

Returns a value note ID for a given value note.

Parameters `value_note` – Value note as string

Returns ID used in Fingertips to represent the value note as integer

1.4.2 api_calls.py

A group of functions to query the Fingertips api and retrieve data in a variety of formats.

`fingertips_py.api_calls.deal_with_url_error(url)`

Parameters `url` – A url that returns a URL Error based on SSL errors

Returns A dataframe from the URL with `verify` set to false.

`fingertips_py.api_calls.get_data_in_tuple(url)`

Parameters `url` – A url to make a request

Returns A tuple of returned data

`fingertips_py.api_calls.get_json(url)`

Parameters `url` – A url to make a request

Returns A parsed JSON object

`fingertips_py.api_calls.get_json_return_df(url, transpose=True)`

Parameters

- `url` – A url to make a request
- `transpose` – [OPTIONAL] transposes dataframe. Default True.

Returns Dataframe generated from JSON response.

`fingertips_py.api_calls.make_request(url, attr=None)`

Parameters

- `url` – A url to make a request
- `attr` – The attribute that needs to be returned

Returns a dict of the attribute and associated data

1.4.3 retrieve_data.py

A group of functions to retrieve data from Fingertips by indicator, profile, domain (group), or geography.

`fingertips_py.retrieve_data.get_all_areas_for_all_indicators()`

Returns a dataframe of all indicators and their geographical breakdowns.

Returns Dataframe of all indicators and their geographical breakdowns

`fingertips_py.retrieve_data.get_all_data_for_indicators(indicators, area_type_id, parent_area_type_id=15, filter_by_area_codes=None)`

Returns a dataframe of data for given indicators at an area.

Parameters

- **indicators** – List or integer or string of indicator Ids
- **area_type_id** – ID of area type (eg. ID of General Practice is 7 etc) used in Fingertips as integer or string
- **parent_area_type_id** – Area type of parent area - defaults to England value
- **filter_by_area_codes** – Option to limit returned data to areas. Areas as either string or list of strings

Returns Dataframe of data for given indicators at an area

`fingertips_py.retrieve_data.get_all_data_for_profile(profile_id, parent_area_type_id=15, area_type_id=None, filter_by_area_codes=None)`

Returns a dataframe of data for all indicators within a profile.

Parameters

- **profile_id** – ID used in Fingertips to identify a profile as integer or string
- **parent_area_type_id** – Area type of parent area - defaults to England value
- **area_type_id** – Option to only return data for a given area type. Area type ids are string, int or a list.
- **filter_by_area_codes** – Option to limit returned data to areas. Areas as either string or list of strings.

Returns A dataframe of data for all indicators within a profile with any filters applied

`fingertips_py.retrieve_data.get_data_by_indicator_ids(indicator_ids, area_type_id, parent_area_type_id=15, profile_id=None, include_sortable_time_periods=None)`

Returns a dataframe of indicator data given a list of indicators and area types.

Parameters

- **indicator_ids** – List of indicator IDs as strings
- **area_type_id** – ID of area type (eg. CCG, Upper Tier Local Authority) used in Fingertips as integer or string
- **parent_area_type_id** – Area type of parent area - defaults to England value
- **profile_id** – ID of profile to select by as either int or string

- **include_sortable_time_periods** – Boolean as to whether to include a sort-friendly data field

Returns A dataframe of data relating to the given indicators

`fingertips_py.retrieve_data.get_data_for_indicator_at_all_available_geographies` (*indicator_id*)
Returns a dataframe of all data for an indicator for all available geographies.

Parameters *indicator_id* – Indicator id

Returns Dataframe of data for indicator for all available areas for all time periods

1.4.4 area_data.py

Functions to retrieve data that are specific to areas and relevant to all indicators. For example: Deprivation decile.

`fingertips_py.area_data.defined_qcut` (*df, value_series, number_of_bins, bins_for_extras, labels=False*)

Allows users to define how values are split into bins when clustering.

Parameters

- **df** – Dataframe of values
- **value_series** – Name of value column to rank upon
- **number_of_bins** – Integer of number of bins to create
- **bins_for_extras** – Ordered list of bin numbers to assign uneven splits
- **labels** – Optional. Labels for bins if required

Returns A dataframe with a new column ‘bins’ which contains the cluster numbers

`fingertips_py.area_data.deprivation_decile` (*area_type_id, area_code=None, year='2015'*)

Takes in an area type id and returns a pandas series of deprivation deciles for those areas (with the areas as an index. If a specific area is requested, it returns just the deprivation decile value.

Parameters

- **area_type_id** – Area type id as denoted by the Fingertips API
- **year** – Year of deprivation score
- **area_code** – Optional. Area code for area type to return a single value for that area

Returns A pandas series of deprivation scores with area codes as the index. Or single value if area is specified.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

f

fingertips_py.api_calls, 5
 fingertips_py.area_data, 7
 fingertips_py.metadata, 2
 fingertips_py.retrieve_data, 5

D

deal_with_url_error() (in module *fingertips_py.api_calls*), 5
 defined_qcut() (in module *fingertips_py.area_data*), 7
 deprivation_decile() (in module *fingertips_py.area_data*), 7

F

fingertips_py.api_calls (module), 5
fingertips_py.area_data (module), 7
fingertips_py.metadata (module), 2
fingertips_py.retrieve_data (module), 5

G

get_age_from_id() (in module *fingertips_py.metadata*), 2
 get_age_id() (in module *fingertips_py.metadata*), 2
 get_all_ages() (in module *fingertips_py.metadata*), 2
 get_all_areas() (in module *fingertips_py.metadata*), 2
 get_all_areas_for_all_indicators() (in module *fingertips_py.retrieve_data*), 6
 get_all_data_for_indicators() (in module *fingertips_py.retrieve_data*), 6
 get_all_data_for_profile() (in module *fingertips_py.retrieve_data*), 6
 get_all_profiles() (in module *fingertips_py.metadata*), 2
 get_all_sexes() (in module *fingertips_py.metadata*), 3
 get_all_value_notes() (in module *fingertips_py.metadata*), 3
 get_area_type_ids_for_profile() (in module *fingertips_py.metadata*), 3
 get_area_types_as_dict() (in module *fingertips_py.metadata*), 3

get_area_types_for_profile() (in module *fingertips_py.metadata*), 3
 get_areas_for_area_type() (in module *fingertips_py.metadata*), 3
 get_data_by_indicator_ids() (in module *fingertips_py.retrieve_data*), 6
 get_data_for_indicator_at_all_available_geographies() (in module *fingertips_py.retrieve_data*), 7
 get_data_in_tuple() (in module *fingertips_py.api_calls*), 5
 get_domains_in_profile() (in module *fingertips_py.metadata*), 3
 get_json() (in module *fingertips_py.api_calls*), 5
 get_json_return_df() (in module *fingertips_py.api_calls*), 5
 get_metadata() (in module *fingertips_py.metadata*), 3
 get_metadata_for_all_indicators() (in module *fingertips_py.metadata*), 3
 get_metadata_for_all_indicators_from_csv() (in module *fingertips_py.metadata*), 4
 get_metadata_for_domain_as_dataframe() (in module *fingertips_py.metadata*), 4
 get_metadata_for_indicator() (in module *fingertips_py.metadata*), 4
 get_metadata_for_indicator_as_dataframe() (in module *fingertips_py.metadata*), 4
 get_metadata_for_profile_as_dataframe() (in module *fingertips_py.metadata*), 4
 get_multiplier_and_calculation_for_indicator() (in module *fingertips_py.metadata*), 4
 get_profile_by_id() (in module *fingertips_py.metadata*), 4
 get_profile_by_name() (in module *fingertips_py.metadata*), 4
 get_sex_from_id() (in module *fingertips_py.metadata*), 5
 get_sex_id() (in module *fingertips_py.metadata*), 5
 get_value_note_id() (in module *fingertips_py.metadata*), 5

M

`make_request ()` (in module `fingertips_py.api_calls`),
5