

---

**fingertips<sub>py</sub>**  
***Release 0.2***

**Sep 02, 2021**



---

## Contents:

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
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 . . . . .	6
1.4.3	retrieve_data.py . . . . .	7
1.4.4	area_data.py . . . . .	8
<b>2</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>
	<b>Index</b>	<b>15</b>



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, is_test=False)
```

Returns an age name from given id.

#### Parameters

- **age\_id** – Age id used in Fingertips as an integer
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** Age or age range as a string

```
fingertips_py.metadata.get_age_id(age, is_test=False)
```

Returns an ID for a given age.

#### Parameters

- **age** – Search term of an age or age range as a string
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

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

fingertips<sub>py</sub>.metadata.**get\_all\_ages** (*is\_test=False*)

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

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** Age codes used in Fingertips in a tuple

fingertips<sub>py</sub>.metadata.**get\_all\_areas** (*is\_test=False*)

Retrieves all area types.

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A dictionary of all area types used in Fingertips

fingertips<sub>py</sub>.metadata.**get\_all\_profiles** (*is\_test=False*)

Returns all profiles.

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

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

fingertips<sub>py</sub>.metadata.**get\_all\_sexes** (*is\_test=False*)

Returns a tuple of all sex categories and their IDs.

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

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

fingertips<sub>py</sub>.metadata.**get\_all\_value\_notes** (*is\_test=False*)

Returns a dictionary of all value notes and their IDs.

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

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

fingertips<sub>py</sub>.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<sub>py</sub>.metadata.**get\_area\_types\_as\_dict** (*is\_test=False*)

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

**Parameters** *is\_test* – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A dictionary of area types

fingertips<sub>py</sub>.metadata.**get\_area\_types\_for\_profile** (*profile\_id, is\_test=False*)

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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A list of dictionaries of area types with relevant information

fingertips<sub>py</sub>.metadata.**get\_areas\_for\_area\_type** (*area\_type\_id*, *is\_test=False*)

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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

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

fingertips<sub>py</sub>.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<sub>py</sub>.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<sub>py</sub>.metadata.**get\_metadata\_for\_all\_indicators** (*include\_definition='no'*, *include\_system\_content='no'*, *is\_test=False*)

Returns the metadata for all indicators in a dataframe.

**Parameters**

- **include\_definition** – optional to include definitions
- **include\_system\_content** – optional to include system content
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** dataframe of all indicators

fingertips<sub>py</sub>.metadata.**get\_metadata\_for\_all\_indicators\_from\_csv** (*is\_test=False*)

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

**Parameters** **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A dataframe of all metadata for all indicators

fingertips<sub>py</sub>.metadata.**get\_metadata\_for\_domain\_as\_dataframe** (*group\_ids*, *is\_test=False*)

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*, *is\_test=False*)  
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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A dictionary of metadata for the given indicator

`fingertips_py.metadata.get_metadata_for_indicator_as_dataframe` (*indicator\_ids*, *is\_test=False*)  
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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A tuple of multiplier and calculation method from Fingertips metadata

`fingertips_py.metadata.get_profile_by_id` (*profile\_id*, *is\_test=False*)  
Returns a profile as an dictionary which contains information about domains and sequencing.

**Parameters**

- **profile\_id** – ID used in Fingertips to identify a profile as integer or string
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** A dictionary of information about the profile

`fingertips_py.metadata.get_profile_by_key` (*profile\_key*)  
Returns a profile object given a key (as the stub following ‘profile’ in the website URL). For example, give, a URL of the form `https://fingertips.phe.org.uk/profile/general-practice/data#page/3/gid/2000...`, the key is ‘general-practice’.

**Parameters** `profile_key` – The exact key for the profile.

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

fingertips<sub>py</sub>.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<sub>py</sub>.metadata.**get\_sex\_from\_id** (*sex\_id, is\_test=False*)

Returns a sex name given an id.

**Parameters**

- **sex\_id** – ID used in Fingertips to represent the sex as integer
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** Sex category as string

fingertips<sub>py</sub>.metadata.**get\_sex\_id** (*sex, is\_test=False*)

Returns an ID for a given sex.

**Parameters**

- **sex** – Sex category as string (Case sensitive)
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** ID used in Fingertips to represent the sex as integer

fingertips<sub>py</sub>.metadata.**get\_value\_note\_id** (*value\_note, is\_test=False*)

Returns a value note ID for a given value note.

**Parameters**

- **value\_note** – Value note as string
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**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<sub>py</sub>.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 varify set to false.

fingertips<sub>py</sub>.api\_calls.**get\_data\_in\_tuple** (*url*)

**Parameters** **url** – A url to make a request

**Returns** A tuple of returned data

fingertips<sub>py</sub>.api\_calls.**get\_json** (*url*)

**Parameters** **url** – A url to make a request

**Returns** A parsed JSON object

fingertips<sub>py</sub>.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<sub>py</sub>.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<sub>py</sub>.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<sub>py</sub>.retrieve\_data.get\_all\_data\_for\_indicators(indicators, area\_type\_id, parent\_area\_type\_id=15, filter\_by\_area\_codes=None, is\_test=False)

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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**Returns** Dataframe of data for given indicators at an area

fingertips<sub>py</sub>.retrieve\_data.get\_all\_data\_for\_profile(profile\_id, parent\_area\_type\_id=15, area\_type\_id=None, filter\_by\_area\_codes=None, is\_test=False)

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.
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**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,  
                                                    is_test=False)
```

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
- **is\_test** – Used for testing. Returns a tuple of expected return and the URL called to retrieve the data

**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, year='2015',  
                                             area_code=None)
```

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, 6  
 fingertips\_py.area\_data, 8  
 fingertips\_py.metadata, 2  
 fingertips\_py.retrieve\_data, 7



## D

deal\_with\_url\_error() (in module *fingertips\_py.api\_calls*), 6  
 defined\_qcut() (in module *fingertips\_py.area\_data*), 8  
 deprivation\_decile() (in module *fingertips\_py.area\_data*), 8

## F

fingertips\_py.api\_calls (module), 6  
 fingertips\_py.area\_data (module), 8  
 fingertips\_py.metadata (module), 2  
 fingertips\_py.retrieve\_data (module), 7

## 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*), 3  
 get\_all\_areas\_for\_all\_indicators() (in module *fingertips\_py.retrieve\_data*), 7  
 get\_all\_data\_for\_indicators() (in module *fingertips\_py.retrieve\_data*), 7  
 get\_all\_data\_for\_profile() (in module *fingertips\_py.retrieve\_data*), 7  
 get\_all\_profiles() (in module *fingertips\_py.metadata*), 3  
 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*), 8  
 get\_data\_for\_indicator\_at\_all\_available\_geographies() (in module *fingertips\_py.retrieve\_data*), 8  
 get\_data\_in\_tuple() (in module *fingertips\_py.api\_calls*), 6  
 get\_domains\_in\_profile() (in module *fingertips\_py.metadata*), 4  
 get\_json() (in module *fingertips\_py.api\_calls*), 6  
 get\_json\_return\_df() (in module *fingertips\_py.api\_calls*), 6  
 get\_metadata() (in module *fingertips\_py.metadata*), 4  
 get\_metadata\_for\_all\_indicators() (in module *fingertips\_py.metadata*), 4  
 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*), 5  
 get\_metadata\_for\_indicator\_as\_dataframe() (in module *fingertips\_py.metadata*), 5  
 get\_metadata\_for\_profile\_as\_dataframe() (in module *fingertips\_py.metadata*), 5  
 get\_multiplier\_and\_calculation\_for\_indicator() (in module *fingertips\_py.metadata*), 5  
 get\_profile\_by\_id() (in module *fingertips\_py.metadata*), 5  
 get\_profile\_by\_key() (in module *fingertips\_py.metadata*), 5  
 get\_profile\_by\_name() (in module *fingertips\_py.metadata*), 6  
 get\_sex\_from\_id() (in module *fingertips\_py.metadata*), 6  
 get\_sex\_id() (in module *fingertips\_py.metadata*), 6

`get_value_note_id()` (*in module `fingertips_py.metadata`*), 6

## M

`make_request()` (*in module `fingertips_py.api_calls`*), 7