# worldometer

*Release 2.0.0*

**Matheus Felipe**

**Jan 16, 2024**

# CONTENTS


*Get live, population, geography, projected, and historical data from around the world.*

CHAPTER

ONE

# ABOUT

The worldometer package accesses various counters and live data available throughout the worldometers.info website and provides them through simple and self-describing classes, methods and attributes.

Access data on:

- The world
- Population
- Geography
- Projections
- Historical

CHAPTER

TWO

# INSTALL

Use `pip` to install the worldometer package:

```
$ pip install worldometer
```

CHAPTER

THREE

# API REFERENCE

The worldometer package is divided into several specific sub-packages for each dataset. So if you are looking for information on a specific package, class, or method, this part of the documentation is for you.

## 3.1 worldometer package

The *worldometer* package accesses various counters and live data available throughout the https://www.worldometers.info website and provides them through self-describing classes, methods and attributes.

### Examples

Get the data from the live counters available on the homepage:

```
>>> from worldometer.world import WorldCounters

>>> wc = WorldCounters()

>>> wc.world_population.current_population
8065299074

>>> wc.government_and_economics.computers_produced_this_year
180248430

>>> wc.society_and_media.internet_users_in_the_world_today
5895566559
```

Reload data to get the latest:

```
>>> wc.reload_data()
>>> wc.world_population.current_population
8065300592
```

Get help and view information about mapped sections:

```
>>> help(wc)
```

### Notes

Check https://www.worldometers.info/about for more information about the data source, how live counters work, and more related information.

## 3.2 world package

**Index**



*world* is the main API of the *worldometer* package, where you can access various live counters and data available on the https://www.worldometers.info website through self-descriptive classes, methods, and attributes.

### 3.2.1 Live Counters

Get the data provided by the live counters.

Each section is represented by an attribute in the *WorldCounters* class, and these attributes are instances of data classes that store the counter values.

Each of these data classes has attributes that describe the data stored in them.

```
>>> from worldometer.world import WorldCounters

>>> wc = WorldCounters()

>>> wc.world_population.current_population
8065299074

>>> wc.government_and_economics.computers_produced_this_year
180248430

>>> wc.society_and_media.internet_users_in_the_world_today
5895566559
```

**class** worldometer.world.counters.**WorldCounters**

Contains a reference to each section of the home page counters.

**source_path**

The source path for the counters.

**Type**
str

**world_population**

An instance of the *WorldPopulation* class that stores all counters related to population data.

**Type**
*WorldPopulation*

**government_and_economics**

An instance of the *GovernmentAndEconomics* class that stores all counters related to government and economic data.

**Type**
*GovernmentAndEconomics*

**society_and_media**

An instance of the *SocietyAndMedia* class that stores all counters related to society and media data.

**Type**
*SocietyAndMedia*

**environment**

An instance of the *Environment* class that stores all counters related to environmental data.

**Type**
*Environment*

**food**

An instance of the *Food* class that stores all counters related to food data.

**Type**
*Food*

**water**

An instance of the *Water* class that stores all counters related to water data.

**Type**
*Water*

**energy**

An instance of the *Energy* class that stores all counters related to energy data.

**Type**
*Energy*

**health**

An instance of the *Health* class that stores all counters related to health data.

**Type**
*Health*

### Notes

For precise and up-to-date information on each section and its counters, please check the worldometers homepage.

**class** worldometer.world.counters.**WorldPopulation**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to world population data.

**current_population**

**Type**
Union[int, float, None]

**births_today**

**Type**
Union[int, float, None]

births_this_year

Type
Union[int, float, None]

deaths_today

Type
Union[int, float, None]

deaths_this_year

Type
Union[int, float, None]

net_population_growth_today

Type
Union[int, float, None]

net_population_growth_this_year

Type
Union[int, float, None]

class worldometer.world.counters.GovernmentAndEconomics(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to government and economic data.

public_healthcare_expenditure_today

Type
Union[int, float, None]

public_education_expenditure_today

Type
Union[int, float, None]

public_military_expenditure_today

Type
Union[int, float, None]

cars_produced_this_year

Type
Union[int, float, None]

bicycles_produced_this_year

Type
Union[int, float, None]

computers_produced_this_year

Type
Union[int, float, None]

class worldometer.world.counters.SocietyAndMedia(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to society and media data.

**new_book_titles_published_this_year**

**Type**
Union[int, float, None]

**newspapers_circulated_today**

**Type**
Union[int, float, None]

**tv_sets_sold_worldwide_today**

**Type**
Union[int, float, None]

**cellular_phones_sold_today**

**Type**
Union[int, float, None]

**money_spent_on_videogames_today**

**Type**
Union[int, float, None]

**internet_users_in_the_world_today**

**Type**
Union[int, float, None]

**emails_sent_today**

**Type**
Union[int, float, None]

**blog_posts_written_today**

**Type**
Union[int, float, None]

**tweets_sent_today**

**Type**
Union[int, float, None]

**google_searches_today**

**Type**
Union[int, float, None]

**class** worldometer.world.counters.**Environment**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to environmental data.

**forest_loss_this_year**

**Type**
Union[int, float, None]

**land_lost_to_soil_erosion_this_year**

**Type**
Union[int, float, None]

co2_emissions_this_year

**Type**
Union[int, float, None]

desertification_this_year

**Type**
Union[int, float, None]

toxic_chemicals_released_in_the_environment_this_year

**Type**
Union[int, float, None]

class worldometer.world.counters.**Food**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to food data.

undernourished_people_in_the_world

**Type**
Union[int, float, None]

overweight_people_in_the_world

**Type**
Union[int, float, None]

obese_people_in_the_world

**Type**
Union[int, float, None]

people_who_died_of_hunger_today

**Type**
Union[int, float, None]

money_spent_for_obesity_related_diseases_in_the_usa_today

**Type**
Union[int, float, None]

money_spent_on_weight_loss_programs_in_the_usa_today

**Type**
Union[int, float, None]

class worldometer.world.counters.**Water**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to water data.

water_used_this_year

**Type**
Union[int, float, None]

deaths_caused_by_water_related_diseases_this_year

**Type**
Union[int, float, None]

people_with_no_access_to_a_safe_drinking_water_source

**Type**
Union[int, float, None]

class worldometer.world.counters.**Energy**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to energy data.

energy_used_today

**Type**
Union[int, float, None]

non_renewable_sources

**Type**
Union[int, float, None]

renewable_sources

**Type**
Union[int, float, None]

solar_energy_striking_earth_today

**Type**
Union[int, float, None]

oil_pumped_today

**Type**
Union[int, float, None]

oil_left

**Type**
Union[int, float, None]

days_to_the_end_of_oil

**Type**
Union[int, float, None]

natural_gas_left

**Type**
Union[int, float, None]

days_to_the_end_of_natural_gas

**Type**
Union[int, float, None]

coal_left

**Type**
Union[int, float, None]

days_to_the_end_of_coal

**Type**
Union[int, float, None]

class worldometer.world.counters.**Health**(*_data: dataclasses.InitVar[Dict[str, Union[int, float, NoneType]]]*)

Counters related to health data.

**communicable_disease_deaths_this_year**

**Type**
Union[int, float, None]

**seasonal_flu_deaths_this_year**

**Type**
Union[int, float, None]

**deaths_of_children_under_5_this_year**

**Type**
Union[int, float, None]

**abortions_this_year**

**Type**
Union[int, float, None]

**deaths_of_mothers_during_birth_this_year**

**Type**
Union[int, float, None]

**hiv_aids_infected_people**

**Type**
Union[int, float, None]

**deaths_caused_by_hiv_aids_this_year**

**Type**
Union[int, float, None]

**deaths_caused_by_cancer_this_year**

**Type**
Union[int, float, None]

**deaths_caused_by_malaria_this_year**

**Type**
Union[int, float, None]

**cigarettes_smoked_today**

**Type**
Union[int, float, None]

**deaths_caused_by_smoking_this_year**

**Type**
Union[int, float, None]

**deaths_caused_by_alcohol_this_year**

**Type**
Union[int, float, None]

**suicides_this_year**

**Type**
Union[int, float, None]

**money_spent_on_illegal_drugs_this_year**

**Type**
Union[int, float, None]

**road_traffic_accident_fatalities_this_year**

**Type**
Union[int, float, None]

### 3.2.2 Country Codes

All countries have specific codes that represent them in some way. Get all of these codes with the *CountryCodes* class:

```
>>> from worldometer.world import CountryCodes

>>> cc = CountryCodes()

>>> cc.data[0]
CountryCodesData(
    country='Afghanistan',
    calling_code='93',
    three_letter_iso='AF',
    two_letter_iso='AFG',
    three_digit_iso_numeric=4
)
```

**class** worldometer.world.country_codes.**CountryCodes**

Represents the data table of some codes used by each country.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

### Notes

Check the source table in Worldometers Country Codes.

**property data: List[*CountryCodesData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.country_codes.**CountryCodesData**(*country: str*, *calling_code: str*, *three_letter_iso: str*, *two_letter_iso: str*, *three_digit_iso_numeric: str*)

Represents a data row from the respective table.

**country**

**Type**
str

**calling_code**

**Type**
str

**three_letter_iso**

**Type**
str

**two_letter_iso**

**Type**
str

**three_digit_iso_numeric**

**Type**
str

## 3.3 population package

**Index**

The population package provides access to various data related to world populations, regions, and their countries. Additionally, it includes historical data ranging from 1950 to the present day and future projections extending to 2050.

To understand the significance of each of these data sets, their sources, and any other related information, please visit the official page at https://www.worldometers.info/population

### 3.3.1 Population by country

```
>>> from worldometer.world.population import CountriesByPopulation

>>> cp = CountriesByPopulation()

>>> cp.data[0]
CountriesByPopulationData(
    idx=1,
    country='India',
    population=1428627663,
    yearly_change='0.81 %',
    net_change=11454490,
    density=481,
    land_area=2973190,
    migrants=-486136,
    fertility_rate=2.0,
    median_age=28.0,
    urban_population='36 %',
    world_share='17.76 %'
)
```

**class** worldometer.world.population.countries_by_population.**CountriesByPopulation**

Represents the data table of countries in the world by population.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

#### Notes

Check the source table in Countries in the world by population.

**property data: List[*CountriesByPopulationData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.countries_by_population.**CountriesByPopulationData**(*idx: int*, *country: str*, *population: int*, *yearly_change: str*, *net_change: int*, *density: int*, *land_area: int*, *migrants: int*, *fertility_rate: float*, *median_age: float*, *urban_population: str*, *world_share: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**yearly_change**

**Type**
str

**net_change**

**Type**
int

**density**

**Type**
int

**land_area**

**Type**
int

**migrants**

**Type**
int

**fertility_rate**

**Type**
float

**median_age**

**Type**
float

**urban_population**

**Type**
str

**world_share**

**Type**
str

### 3.3.2 Population by region

```
>>> from worldometer.world.population import WorldPopulationByRegion

>>> pr = WorldPopulationByRegion()

>>> pr.current()[0]
CurrentWorldPopulationByRegionData(
    idx=1,
    region='Asia',
    population=4753079727,
    yearly_change='0.64 %',
    net_change=30444963,
    density=153,
    area=31033131,
    migrants=-1487191,
    fertility_rate=1.934,
    median_age=32,
    urban_population='52.6 %',
    world_share='59.1 %'
)
```

(continues on next page)

(continued from previous page)

```
>>> pr.past()[0]
PastWorldPopulationByRegionData(
    idx=1,
    region='Asia',
    population=1379048370,
    world_share='55.2 %'
)

>>> pr.future()[0]
FutureWorldPopulationByRegionData(
    idx=1,
    region='Asia',
    population=5292947571,
    world_share='54.5 %'
)
```

**class** worldometer.world.population.by_region.**WorldPopulationByRegion**

Represents the data table of regions in the world by population.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

### Notes

Check the source table in Regions in the world by population.

**current**() → List[*CurrentWorldPopulationByRegionData*]

Get a list of all the current data from the table.

These data are related to the current year.

Each index in the list contains an object representing a data row of the table.

**future**() → List[*FutureWorldPopulationByRegionData*]

Get a list of all future data from the table.

These data are an estimate for the year 2050.

Each index in the list contains an object representing a data row of the table.

**past**() → List[*PastWorldPopulationByRegionData*]

Get a list of all historical data from the table.

These data pertain to the year 1950.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.by_region.**CurrentWorldPopulationByRegionData**(*idx: int*, *region: str*, *population: int*, *yearly_change: str*, *net_change: int*, *density: int*, *area: int*, *migrants: int*, *fertility_rate: float*, *median_age: int*, *urban_population: str*, *world_share: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**region**

**Type**
str

**population**

**Type**
int

**yearly_change**

**Type**
str

**net_change**

**Type**
int

**density**

**Type**
int

**area**

**Type**
int

**migrants**

**Type**
int

`fertility_rate`

**Type**
float

`median_age`

**Type**
int

`urban_population`

**Type**
str

`world_share`

**Type**
str

class worldometer.world.population.by_region.**PastWorldPopulationByRegionData**(*idx: int*, *region: str*, *population: int*, *world_share: str*)

Represents a data row from the respective table.

`idx`

**Type**
int

`region`

**Type**
str

`population`

**Type**
int

`world_share`

**Type**
str

class worldometer.world.population.by_region.**FutureWorldPopulationByRegionData**(*idx: int*, *region: str*, *population: int*, *world_share: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**region**

**Type**
str

**population**

**Type**
int

**world_share**

**Type**
str

### 3.3.3 Population by year

```
>>> from worldometer.world.population import WorldPopulationByYear

>>> py = WorldPopulationByYear()

>>> py.data[0]
WorldPopulationByYearData(
    year=2023,
    world_population=8045311447,
    yearly_change='0.88 %',
    net_change=70206291.0,
    density=54.0
)
```

**class** worldometer.world.population.by_year.**WorldPopulationByYear**

Represents the data table of the world population by year.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

#### Notes

Check the source table in the world population by year.

**property data: List[*WorldPopulationByYearData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.population.by_year.**WorldPopulationByYearData**(*year: int*, *world_population: int*, *yearly_change: str*, *net_change: float*, *density: float*)

Represents a data row from the respective table.

**year**

**Type**
int

**world_population**

**Type**
int

**yearly_change**

**Type**
str

**net_change**

**Type**
float

**density**

**Type**
float

### 3.3.4 Largest cities in the world

```
>>> from worldometer.world.population import LargestCities

>>> lc = LargestCities()

>>> lc.data[0]
LargestCitiesData(
    rank=1,
    urban_area='Tokyo-Yokohama',
    population_estimate=37843000,
    country='Japan',
    land_area=8547,
    density=4400
)
```

class worldometer.world.population.largest_cities.**LargestCities**

Represents the data table of the largest cities in the world.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

### Notes

Check the source table in the largest cities in the world.

**property data: List[*LargestCitiesData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.largest_cities.**LargestCitiesData**(*rank: int*, *urban_area: str*, *population_estimate: str*, *country: str*, *land_area: int*, *density: int*)

Represents a data row from the respective table.

**rank**

**Type**
int

**urban_area**

**Type**
str

**population_estimate**

**Type**
str

**country**

**Type**
str

**land_area**

**Type**
int

**density**

**Type**
int

### 3.3.5 Most populous countries

```
>>> from worldometer.world.population import MostPopulousCountries

>>> pc = MostPopulousCountries()

>>> pc.current()[0]
CurrentMostPopulousCountriesData(
    idx=1,
    country='India',
    population=1428627663,
    yearly_change='0.81 %',
    world_share='17.8 %'
)

>>> pc.past()[0]
PastMostPopulousCountriesData(
    idx=1,
    country='China',
    population=543979233,
    world_share='21.8 %',
    rank='(2)'
)

>>> pc.future()[0]
FutureMostPopulousCountriesData(
    idx=1,
    country='India',
    population=1670490596,
    world_share='17.2 %',
    rank='(1)'
)
```

**class** worldometer.world.population.most_populous_countries.**MostPopulousCountries**

Represents the data table of most populous countries in the world.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

### Notes

Check the source table in Most populous countries in the world.

**current**() → List[*CurrentMostPopulousCountriesData*]

Get a list of all the current data from the table.

These data are related to the current year.

Each index in the list contains an object representing a data row of the table.

**future**() → List[*FutureMostPopulousCountriesData*]

Get a list of all future data from the table.

These data are an estimate for the year 2050.

Each index in the list contains an object representing a data row of the table.

**past**() → List[*PastMostPopulousCountriesData*]

Get a list of all historical data from the table.

These data pertain to the year 1950.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.population.most_populous_countries.**CurrentMostPopulousCountriesData**(*idx: int*, *country: str*, *population: int*, *yearly_change: str*, *world_share: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**yearly_change**

**Type**
str

**world_share**

**Type**
str

class worldometer.world.population.most_populous_countries.**PastMostPopulousCountriesData**(*idx: int*, *country: str*, *population: int*, *world_share: str*, *rank: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**world_share**

**Type**
str

**rank**

**Type**
str

class worldometer.world.population.most_populous_countries.**FutureMostPopulousCountriesData**(*idx: int*, *country: str*, *population: int*, *world_share: str*, *rank: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**world_share**

**Type**
str

**rank**

**Type**
str

### 3.3.6 World population projections

```
>>> from worldometer.world.population import WorldPopulationProjections

>>> pp = WorldPopulationProjections()

>>> pp.data[0]
WorldPopulationProjectionsData(
    year=2023,
    world_population=8045311447,
    yearly_change='0.88 %',
    net_change=70206291,
    density=54
)
```

**class** worldometer.world.population.projections.**WorldPopulationProjections**

Represents the data table of the world population projections.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

#### Notes

Check the source table in World Population Projections.

**property data: List[*WorldPopulationProjectionsData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.population.projections.**WorldPopulationProjectionsData**(*year: int*, *world_population: int*, *yearly_change: str*, *net_change: int*, *density: int*)

Represents a data row from the respective table.

**year**

**Type**
int

**world_population**

**Type**
int

**yearly_change**

**Type**
str

**net_change**

**Type**
int

**density**

**Type**
int

### 3.3.7 Regions population

```
>>> from worldometer.world.population import (
        AsiaPopulation,
        AfricaPopulation,
        EuropePopulation,
        LatinAmericanAndTheCaribbeanPopulation,
        NorthernAmericanPopulation,
        OceaniaPopulation
    )

>>> ap = AsiaPopulation()
```

(continues on next page)

(continued from previous page)

```
>>> ap.live()
4762699828

>>> ap.subregions()[0]
SubregionData(
    area='Southern Asia',
    population='(2,027,578,876)'
)

>>> ap.historical()[0]
HistoricalData(
    year=2023,
    population=4753079727,
    yearly_change_percent='0.64 %',
    yearly_change=30444963,
    migrants=-1487191,
    median_age=31.9,
    fertility_rate=1.93,
    density=153,
    urban_population_percent='52.6 %',
    urban_population=2500201501,
    world_share='59.1 %',
    world_population=8045311447,
    rank=nan
)

>>> ap.forecast()[0]
ForecastData(
    year=2025,
    population=4816249054,
    yearly_change_percent='0.64 %',
    yearly_change=30384996,
    migrants=-1555419,
    median_age=32.7,
    fertility_rate=1.93,
    density=155,
    urban_population_percent='53.8 %',
    urban_population=2589655469,
    world_share='61.4 %',
    world_population=8191988453,
    rank=nan
)
```

**class** worldometer.world.population.regions.**AsiaPopulation**

Represents the data tables of Asia's populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Asia Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.population.regions.**AfricaPopulation**

Represents the data tables of Africa's populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Africa Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.regions.**EuropePopulation**

Represents the data tables of Europe's populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Europe Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.regions.**LatinAmericanAndTheCaribbeanPopulation**

Represents the data tables of Latin American and Caribbean populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Latin America and the Caribbean Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.population.regions.**NorthernAmericanPopulation**

Represents the data tables of Northern American populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Northern American Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.regions.**OceaniaPopulation**

Represents the data tables of the Oceania populations.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

### Notes

Check the source tables in Oceania Population.

**forecast**() → List[*ForecastData*]

Get a list of all forecast data from the table.

Each index in the list contains an object representing a data row of the table.

**historical**() → List[*HistoricalData*]

Get a list of all historical data from the table.

Each index in the list contains an object representing a data row of the table.

**live**() → int | float | None

Get a live population counter for the respective region.

**subregions**() → List[*SubregionData*]

Get a list of all the subregions' data from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.population.regions.**SubregionData**(*area: int*, *population: str*)

Represents a data row from the respective table.

**area**

**Type**
int

**population**

**Type**
str

class worldometer.world.population.regions.**HistoricalData**(*year: int*, *population: int*, *yearly_change_percent: str*, *yearly_change: int*, *migrants: int*, *median_age: float*, *fertility_rate: float*, *density: int*, *urban_population_percent: str*, *urban_population: int*, *world_share: str*, *world_population: int*, *rank: int*)

Represents a data row from the respective table.

`year`

**Type**
int

`population`

**Type**
int

`yearly_change_percent`

**Type**
str

`yearly_change`

**Type**
int

`migrants`

**Type**
int

`median_age`

**Type**
float

`fertility_rate`

**Type**
float

`density`

**Type**
int

`urban_population_percent`

**Type**
str

`urban_population`

**Type**
int

`world_share`

**Type**
str

`world_population`

**Type**
int

**rank**

**Type**
int

class worldometer.world.population.regions.**ForecastData**(*year: int*, *population: int*, *yearly_change_percent: str*, *yearly_change: int*, *migrants: int*, *median_age: float*, *fertility_rate: float*, *density: int*, *urban_population_percent: str*, *urban_population: int*, *world_share: str*, *world_population: int*, *rank: int*)

Represents a data row from the respective table.

**year**

**Type**
int

**population**

**Type**
int

**yearly_change_percent**

**Type**
str

**yearly_change**

**Type**
int

**migrants**

**Type**
int

**median_age**

**Type**
float

**fertility_rate**

**Type**
float

**density**

**Type**
int

**urban_population_percent**

**Type**
str

**urban_population**

**Type**
int

**world_share**

**Type**
str

**world_population**

**Type**
int

**rank**

**Type**
int

## 3.4 geography package

**Index**



The *geography* package provides access to various data related to geographic information about the world, regions, and their countries.

To understand the significance of each of these data sets, their sources, and any other related information, please visit the official page at https://www.worldometers.info/geography/how-many-countries-are-there-in-the-world

### 3.4.1 Countries in the world

```
>>> from worldometer.world.geography import (
        WorldCountries,
        AsiaCountries,
        AfricaCountries,
        EuropeCountries,
        LatinAmericanAndTheCaribbeanCountries,
        NorthernAmericanCountries,
        OceaniaCountries
    )

>>> wc = WorldCountries()

>>> wc.total
195
```

(continues on next page)

(continued from previous page)

```
>>> wc.countries()[0]
WorldCountriesData(
    idx=1,
    country='India',
    population=1428627663,
    world_share='17.76 %',
    land_area=2973190
)

>>> ac = AfricaCountries()

>>> ac.total
54

>>> ac.countries()[0]
CountryData(
    idx=1,
    country='Nigeria',
    population=223804632,
    subregion='Western Africa'
)

>>> ac.dependencies()[0]
DependencyData(
    idx=1,
    territory='Réunion',
    population=981796,
    dependency_of='France'
)
```

**class** worldometer.world.geography.countries.**WorldCountries**

Represents the data table of a list of countries in the world.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

**total**

The total number of countries in the world.

**Type**
int

### Notes

Check the source table in List of countries.

**countries**() → List[*WorldCountriesData*]

Get a list of all the countries' data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**AsiaCountries**

Represents the data tables of Asia's countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Asia.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**AfricaCountries**

Represents the data tables of Africa's countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Africa.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**EuropeCountries**

Represents the data tables of Europe's countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Europe.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**LatinAmericanAndTheCaribbeanCountries**

Represents the data tables of Latin American And The Caribbean countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Latin American And The Caribbean.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

class worldometer.world.geography.countries.**NorthernAmericanCountries**

Represents the data tables of Northern American countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Northern American.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**OceaniaCountries**

Represents the data tables of the Oceania countries.

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original tables.

**Type**
tuple

**total**

The total number of countries in the region.

**Type**
int

### Notes

Check the source tables in Countries in Oceania.

**countries**() → List[*CountryData*]

Get a list of all the data for countries in the region from the table.

Each index in the list contains an object representing a data row of the table.

**dependencies**() → List[*DependencyData*]

Get a list of all the data for dependencies in the region from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.countries.**WorldCountriesData**(*idx: int*, *country: str*, *population: int*, *world_share: str*, *land_area: int*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**world_share**

**Type**
str

**land_area**

**Type**
int

class worldometer.world.geography.countries.**CountryData**(*idx: int*, *country: str*, *population: int*, *subregion: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**country**

**Type**
str

**population**

**Type**
int

**subregion**

**Type**
str

class worldometer.world.geography.countries.**DependencyData**(*idx: int*, *territory: str*, *population: int*, *dependency_of: str*)

Represents a data row from the respective table.

**idx**

**Type**
int

**territory**

**Type**
str

**population**

**Type**
int

**dependency_of**

**Type**
str

### 3.4.2 Largest countries in the world

```
>>> from worldometer.world.geography import LargestCountries

>>> lc = LargestCountries()

>>> lc.data[0]
LargestCountriesData(
    idx=1,
    country='Russia',
    total_area_km2=17098242,
    total_area_mi2=6601665,
    land_area_km2=16376870,
    land_area_mi2=6323142,
    percentage_of_world_landmass='11.0 %'
)
```

**class** worldometer.world.geography.largest_countries.**LargestCountries**

Represents the data table of the largest countries in the world (by area).

**source_path**

The data source path.

**Type**
str

**new_column_names**

The new column names that will be used to replace those of the original table.

**Type**
tuple

#### Notes

Check the source table in the Largest Countries in the World.

**property data: List[*LargestCountriesData*]**

Get a list of all the data from the table.

Each index in the list contains an object representing a data row of the table.

**class** worldometer.world.geography.largest_countries.**LargestCountriesData**(*idx: int*, *country: str*, *total_area_km2: int*, *total_area_mi2: int*, *land_area_km2: int*, *land_area_mi2: int*, *percentage_of_world_landmass: str*)

Represents a data row from the respective table.

`idx`

**Type**
int

`country`

**Type**
str

`total_area_km2`

**Type**
int

`total_area_mi2`

**Type**
int

`land_area_km2`

**Type**
int

`land_area_mi2`

**Type**
int

`percentage_of_world_landmass`

**Type**
str

CHAPTER

# FOUR

# DEMO

**Note:** The first time you run any function/method or class, it will download Chromium to `~/.local/share/pyppeteer` directory. It only happens once. After, it will only open the chromium to render the contents of worldometers.info.

Get the data from the live counters available on the homepage:

```
>>> from worldometer.world import WorldCounters

>>> wc = WorldCounters()

>>> wc.world_population.current_population
8065299074

>>> wc.government_and_economics.computers_produced_this_year
180248430

>>> wc.society_and_media.internet_users_in_the_world_today
5895566559
```

Reload data to get the latest:

```
>>> wc.reload_data()
>>> wc.world_population.current_population
8065300592
```

Get help and view information about mapped sections:

```
>>> help(wc)
```

CHAPTER

FIVE

# WORLDOMETERS.INFO

> "Worldometer is run by an international team of developers, researchers, and volunteers with the goal of making world statistics available in a thought-provoking and time relevant format to a wide audience around the world. It is published by a small and independent digital media company based in the United States. We have no political, governmental, or corporate affiliation. Furthermore, we have no investors, donors, grants, or backers of any type. We are completely independent and self-financed through automated programmatic advertising sold in real time on multiple ad exchanges."

More info: worldometers.info/about

CHAPTER

SIX

# DATA SOURCES

**[adapted]:** "worldometers.info collects its statistics and data from the most reputable national and international organizations, including the United Nations, the World Health Organization, the Food and Agriculture Organization, OECD and others.

Each Worldometer counter has its specific set of sources, which are listed on its dedicated page (accessible by clicking on the counter text link, when available).

Data, estimates, and projections displayed on worldometers.info counters are for the most part provided by organizations included in the following list of United Nations Statistics Division's partners."

More info about data source: worldometers.info/sources

CHAPTER

SEVEN

# CONTRIBUTIONS

All contributions are welcome!

Found a problem, want to give a tip? open an issue

Do you have a solution to the problem? Send me a PR

Did you like this project? Click on the star

CHAPTER

EIGHT

# LICENSE

This project is using the MIT license, see in MIT LICENSE

CHAPTER

NINE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## W

# INDEX

## D

## E



## F



## G



## H

## I



## L



## M

## N

## O



## P



## R

## S

## T



## U



## W

## Y