

Abstract Data Type

Array

Marian Petruk

May 11 2017

Contents

1	Definition(s)	2
2	Array1D	3
3	Array2D	3
4	Indicator ADT class	4
5	Usage	5
6	Indicators	5
7	Example	5

What is an array?



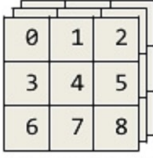
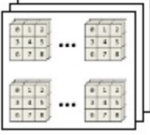
Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3 rd order Tensor)
N		ND Array

Figure 1: Visual representation of an array

1 Definition(s)

1. In computer science, **an array data structure**, or simply **an array**, is a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key. An array is stored so that the position of each element can be computed from its index tuple by a mathematical formula. The simplest type of data structure is a linear array, also called one-dimensional array. - [Wikipedia, the free encyclopedia](#).
2. **An array** keeps track of multiple pieces of information in linear order, a one-dimensional list. However, the data associated with certain systems (a digital image, a board game, etc.) lives in two dimensions. To visualize this data, we need a multi-dimensional data structure, that is, **a multi-dimensional array**. A two-dimensional array is really nothing more than an array of arrays (a three-dimensional array is an array of arrays of arrays). - [Processing.org](#)
3. **The array** is a basic abstract data type that holds an ordered collection of items accessible by an integer index. These items can be anything from primitive types such as integers to more complex types like instances of classes. Since it's an ADT, it doesn't specify an implementation, but is almost always implemented by an array (data structure) or dynamic array. - [Brilliant Math & Science Wiki](#)

2 Array1D

Data:

- Any type

Operation(s):

- `__len__` - Returns the size of the array.
- `__getitem__` - Returns the value of the element on the given index.
- `__setitem__` - Puts (sets) the value in the array's item at the given index position.
- `__clear__` - Clears the array by setting each item to the given value.
- `__str__` - Converts the adt structure of 1D array into a string.
- `__iter__` - Returns the array's iterator for traversing the items.

3 Array2D

Data:

- Any type

Operation(s):

- `num_rows` - Returns the number of rows in the 2D array.
- `num_cols` - Returns the number of columns in the 2D array.
- `clear` - Returns the number of columns in the 2D array.
- `__getitem__` - Returns the number of columns in the 2D array.
- `__setitem__` - Returns the number of columns in the 2D array.
- `__str__` - Converts the 2D array into a string.

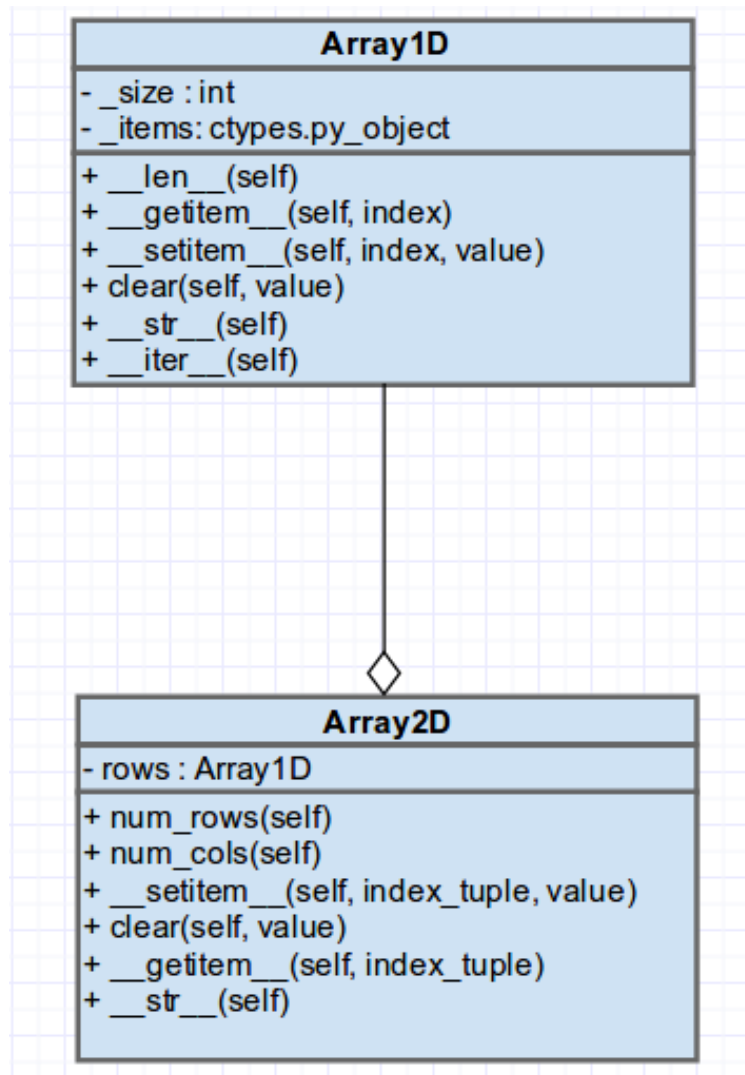


Figure 2: *UML Diagram of Array1D and Array2D classes*

4 Indicator ADT class

Operation(s):

- `__init__` - Creates an array using Array2D data structure. Gets data from the wbpy api. Fills the array with these data.
- `indicator_name` - Returns the name of the indicator.
- `country_codes` - Returns the iso codes of the countries.

- `indicator_api_url` - Returns the url of the api request for the indicator.
- `get_array` - Returns the array with indicator data.
- `print_year` - Returns indicator data of the given country for the given year.
- `array_2_list` - Returns the list with indicator data of the given country from the array.
- `__str__` - For `print()` function. Converts the ADT into str.

5 Usage

In this project, ADT of the 2 dimensional array data structure will be used. In one column there will be a data from a country (e.g. UA), and in other another country (e.g. PL). Rows will be filled with data from The World Bank. Every 2D array will consist data from one indicator.

6 Indicators

```
#                               Indicators                               #
# -----#
#                               Economy and growth                       #
# -----#

gdp = "NY.GDP.MKTP.CD" # GDP (current US$)
gdp_per_capita = "NY.GDP.PCAP.CD" # GDP per capita (current US$)
gross_savings = "NY.GNS.ICTR.ZS" # Gross savings (% of GDP)
inflation_gdp = "NY.GDP.DEFL.KD.ZG" # Inflation, GDP deflator (annual %)
imports = "NE.IMP.GNFS.ZS" # Imports of goods and services (% of GDP)
inflation_consumer_prices = "FP.CPI.TOTL.ZG" # Inflation, consumer prices (annual %)
gni = "NY.GNP.MKTP.PP.CD" # GNI, PPP (current international $)

# -----#
#                               Some other indicators                       #
# -----#

total_population = "SP.POP.TOTL" # total population indicator
life_expectancy = "SP.DYN.LE00.IN" # Life expectancy at birth, total (years)
high_tech_exports = "TX.VAL.TECH.CD" # High-technology exports (current US$)
science_tech_articles = "IP.JRN.ARTC.SC" # Scientific and technical journal articles
```

7 Example

Result of using data structure for total population indicator of Ukraine and Poland from 1990 to 2017 (2016):

```
>>> indicator1 = indicator_adt(total_population)
>>> print("Current indicator:", indicator1.indicator_name())
Current indicator: Population, total
```

```
>>> print("Api URL of the indicator:", indicator1.indicator_api_url())
Api URL of the indicator: http://api.worldbank.org/countries/UA;PL/indicators/SP.POP.TOTL?date=1990%3A2017&format=json&per_page=100
>>> print("Iso codes of countries in the indicator:", indicator1.country_codes())
Iso codes of countries in the indicator: UA PL
>>> print(indicator1)
Indicator Population, total
      UA      PL
1990 year: 51892000.0 38110782.0
1991 year: 52000470.0 38246193.0
1992 year: 52150266.0 38363667.0
1993 year: 52179210.0 38461408.0
1994 year: 51921041.0 38542652.0
1995 year: 51512299.0 38594998.0
1996 year: 51057189.0 38624370.0
1997 year: 50594105.0 38649660.0
1998 year: 50143939.0 38663481.0
1999 year: 49673350.0 38660271.0
2000 year: 49175848.0 38258629.0
2001 year: 48683865.0 38248076.0
2002 year: 48202500.0 38230364.0
2003 year: 47812950.0 38204570.0
2004 year: 47451600.0 38182222.0
2005 year: 47105150.0 38165445.0
2006 year: 46787750.0 38141267.0
2007 year: 46509350.0 38120560.0
2008 year: 46258200.0 38125759.0
2009 year: 46053300.0 38151603.0
2010 year: 45870700.0 38042794.0
2011 year: 45706100.0 38063255.0
2012 year: 45593300.0 38063164.0
2013 year: 45489600.0 38040196.0
2014 year: 45271947.0 38011735.0
2015 year: 45154029.0 37986412.0
```
