# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 11 10:12:52 2019

@author: sugandha.sharma

"""

"""Arrays:

Arrays are used to store multiple values in one single

variable:"""

"""Example:

Create an array containing car names:"""

cars = ["Ford", "Volvo", "BMW"]

"""What is an Array?:

An array is a special variable, which can hold more than

one value at a time.

If you have a list of items (a list of car names,

for example), storing the cars in single variables

could look like this:"""

car1 = "Ford"

car2 = "Volvo"

car3 = "BMW"

"""However, what if you want to loop through the cars

and find a specific one? And what if you had not 3 cars,

but 300?"""

"""The solution is an array!"""

"""An array can hold many values under a single name,

and you can access the values by referring to an index

number."""

"""Access the Elements of an Array:

You refer to an array element by referring to the index

number."""

"""Example"

Get the value of the first array item:"""

x = cars[0]

print(x)

"""Example:

Modify the value of the first array item:"""

x=cars[0]

cars[0] = "Toyota"

print(x)

"""The Length of an Array:

Use the len() method to return the length of an array

(the number of elements in an array)."""

"""Example:

Return the number of elements in the cars array:"""

y = len(cars)

print(y)

"""Note: The length of an array is always one more than

the highest array index."""

"""Looping Array Elements:

You can use the for in loop to loop through all the

elements of an array."""

"""Example:

Print each item in the cars array:"""

for z in cars:

print(z)

"""Adding Array Elements:

You can use the append() method to add an element to

an array."""

"""Example:

Add one more element to the cars array:"""

cars.append("Honda")

print(cars)

"""Removing Array Elements:

You can use the pop() method to remove an element from

the array."""

"""Example:

Delete the second element of the cars array:"""

cars.pop(1)

"""You can also use the remove() method to remove

an element from the array."""

"""Example:

Delete the element that has the value "Volvo":"""

cars.remove("Toyota")

print(cars)

"""

Note: The list's remove() method only removes the first

occurrence of the specified value."""

"""Array Methods:

Python has a set of built-in methods that you can use

on lists/arrays."""

"""Method Description:

append() Adds an element at the end of the list

clear() Removes all the elements from the list

copy() Returns a copy of the list

count() Returns the number of elements with the specified

value

extend() Add the elements of a list (or any iterable),

to the end of the current list

index() Returns the index of the first element with the

specified value

insert() Adds an element at the specified position

pop() Removes the element at the specified

position

remove() Removes the first item with the specified

value

reverse() Reverses the order of the list

sort() Sorts the list

Note: Python does not have built-in support for Arrays,

but Python Lists can be used instead.

"""

array1 =array('i', [10,20,30,40,50])

for x in array1:

print(x)

"""Why use Arrays in Python?

A combination of Arrays, together with Python could

save you a lot of time. As mentioned earlier,

arrays help you reduce the overall size of your

code, while Python helps you get rid of problematic

syntax, unlike other languages.

For example: If you had to store integers from 1-100,

you won’t be able to remember 100 variable names

explicitly, therefore, you can save them easily

using an array.

What is an Array?

An array is basically a data structure which can

hold more than one value at a time.

It is a collection or ordered series of elements of

the same type.

Example:1:"""

a=array('d',[1.2,1.3,2.3])

"""

We can loop through the array items easily and fetch

the required values by just specifying the index

number.

Arrays are mutable(changeable) as well,

therefore, you can perform various manipulations as

required.

Now, there is always a question that comes up to

our mind –

Is Python list same as an Array?

The ‘array’ data structure in core python is not

very efficient or reliable. Therefore, when we talk

about python arrays, we usually mean python lists.

However, python does provide Numpy Arrays which are

a grid of values used in Data Science.

"""

"""Creating an Array:

Arrays in Python can be created after importing the

array module as follows –

→ import array as arr

The array(data type, value list) function takes

two parameters, the first being the data type

of the value to be stored and the second is the

value list. The data type can be anything such as int,

float, double, etc. Please make a note that arr is

the alias name and is for ease of use. You can import

without alias as well. There is another way to import

the array module which is –

→ from array import \*

This means you want to import all functions from

the array module.

The following syntax is used to create an array.

Syntax:

1

a=arr.array(data type,value list) #when you import

using arr alias

OR

1

a=array(data type,value list) #when you import using \*

Example: a=arr.array( ‘d’ , [1.1 , 2.1 ,3.1] )

Here, the first parameter is ‘d’ which is a

data type i.e. float and the values are specified as

the next parameter.

Note:

All values specified are of the type float.

We cannot specify the values of different data

types to a single array.

The following table shows you the various data

types and their codes.

Type code Python Data Type Byte size

i int 2

I int 2

u unicode character 2

h int 2

H int 2

l int 4

L int 4

f float 4

d float 8

Accessing array elements :

To access array elements, you need to specify the

index values. Indexing starts at 0 and not from 1.

Hence, the index number is always 1 less than the length of the array.

Syntax:

Array\_name[index value]

Example:

"""

import array as arr

a=arr.array( 'd', [1.0 , 2.0 ,3.0] )

print(a[1])

"""The output returned is the value, present at the

second place in our array which is 2.1.

Let us have a look at some of the basic array

operations now.

Basic array operations :

There are many operations that can be performed on

arrays which are as follows –

Basic Array Operations - Arrays In Python -

Finding the Length of an Array:

Length of an array is the number of elements that are

actually present in an array.

You can make use of len() function to achieve this.

The len() function returns an integer value that is equal to the number of elements present in that array.

Syntax:

→ len(array\_name)

Example:

"""

a=arr.array('d', [1.1 , 2.1 ,3.1] )

len(a)

"""This returns a value of 3 which is equal to the number

of array elements."""

"""Adding/ Changing elements of an Array:

We can add value to an array by using the append(),

extend() and the insert (i,x) functions.

The append() function is used when we need to add a

single element at the end of the array.

Example:"""

a=arr.array('d', [1.1 , 2.1 ,3.1] )

a.append(3.4)

print(a)

"""The resultant array is the actual array with the

new value added at the end of it. To add more than

one element, you can use the extend() function.

This function takes a list of elements as its

parameter. The contents of this list are the elements

to be added to the array.

Example:"""

a=arr.array('d', [1.1 , 2.1 ,3.1] )

a.extend([4.5,6.3,6.8])

print(a)

"""The resulting array will contain all the 3 new

elements added to the end of the array.

However, when you need to add a specific element

at a particular position in the array, the insert(i,x)

function can be used. This function inserts the

element at the respective index in the array.

It takes 2 parameters where the first parameter

is the index where the element needs to be inserted

and the second is the value.

Example:"""

a=arr.array('d', [1.1 , 2.1 ,3.1] )

a.insert(2,3.8)

print(a)

"""The resulting array contains the value 3.8 at

the 3rd position in the array.

Arrays can be merged as well by performing array

concatenation.

Array Concatenation :

Any two arrays can be concatenated using the + symbol.

Example:"""

a=arr.array('d',[1.0 , 2.0 ,3.0,2.0,7.0])

b=arr.array('d',[3.7,8.6])

c=arr.array('d')

c=a+b

##a=a+b

print("Array c = ",c)

#print("Array a = ",a)

#d=a+b

#print("Array d = ",d)

"""

The resulting array c contains concatenated

elements of arrays a and b."""

"""

Now, let us see how you can remove or delete items

from an array.

Removing/ Deleting elements of an array:

Array elements can be removed using pop() or

remove() method. The difference between these

two functions is that the former returns the

deleted value whereas the latter does not.

The pop() function takes either no parameter or the

index value as its parameter.

When no parameter is given,

this function pops() the last element and returns it.

When you explicitly supply the index value,

the pop() function pops the required elements and

returns it.

Example:"""

a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])

print(a.pop())

print(a.pop(3))

"""The first pop() function removes the last value

4.6 and returns the same while the second one pops

the value at the 4th position which is 3.1 and returns

the same.

The remove() function, on the other hand, is used to

remove the value where we do not need the removed

value to be returned. This function takes the element

value itself as the parameter. If you give the index

value in the parameter slot, it will throw an error.

Example:

"""

a=arr.array('d',[1.1 , 2.1 ,3.1])

a.remove(1.1)

print(a)

"""When you want a specific range of values from an

array, you can slice the array to return the same,

as follows.

"""

"""Slicing an array :

An array can be sliced using the : symbol.

This returns a range of elements that we have

specified by the index numbers.

Example:"""

a=arr.array('d',[1.1 , 2.1 ,3.8,3.7,1.2])

print(a[2:3])

"""The result will be elements present at 1st, 2nd and

3rd position in the array."""

"""Looping through an array:

Using the for loop, we can loop through an array.

Example:"""

a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])

print("All values")

for x in a:

print(x)

print("specific values")

for x in a[1:3]:

print(x)

"""

The above output shows the result using for loop.

When we use for loop without any specific parameters,

the result contains all the elements of the array given

one at a time. In the second for loop, the result

contains only the elements that are specified using

the index values. Please note that the result does not

contain the value at index number 3. """