Arrays

In this lesson, we learn about arrays and how they are used in Python!

We'll cover the following

Introduction#

In Python, an array is just an ordered sequence of *homogeneous* elements. In other words, an array can only hold elements of one datatype. Python arrays are basically just wrappers for C arrays. The type is constrained and specified at the time of creation.

Initializing Arrays#

Python arrays are initialized using the array library:

```
import array
new_array = array.array('type', [list])
```

Here `type` defines the data type of array and `list` is a pyth[illegible] containing homogenous elements.

Consider the example below:

```
1  import array
2
3  # type: 'f' (float), initializer list: [1, 2, 3]
4  new_array = array.array('f', [1, 2, 3])
5  print(new_array[0])
```

Line 4 creates an array. Here `f` indicates that the array is of type `float`.

Types of Arrays#

There are several types of arrays in Python; refer to the table below for a complete list.

Type code	C Type	Python Type	Minimum Size in Bytes
`'c'`	character	char	1
`'b'`	signed char	int	1
`'B'`	unsigned char	int	1

Type code	C Type	Python Type	in Bytes
'u'	Py_UNICODE	Unicode character	2 or 4 depending on Unicode build
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	long	4
'l'	signed long	int	4
'L'	unsigned long	long	4
'f'	float	float	4
'd'	double	float	8

Array slicing

Array slicing is done in exactly the same way as list slicing is done. Look at the following example:

```
import array
```

```
initializer_list = [2, 5, 43, 5, 10, 52, 29, 5]
number_array = array.array('i', initializer_list)

print(number_array[1:5])  # 2nd to 5th
print(number_array[:-5])  # beginning to 3rd
print(number_array[5:])  # 6th to end
print(number_array[:])   # beginning to end
```

Changing or adding array elements#

Arrays are mutable; their elements can be changed in the same way as list elements. Have a look at the following coding widget.

```
import array
integers = array.array('i', [1, 2, 3, 5, 7, 10])

# changing first element
integers[0] = 0
print(integers)  # array('i', [0, 2, 3, 5, 7, 10])

# changing 3rd to 5th element
integers[2:5] = array.array('i', [4, 6, 8])
print(integers)     # Output: array('i', [0, 2, 4, 6, 8, 10])
```

Just as with lists, we can add one item to the end of an array using the `append()` method or add several items using the `extend()` method.

```
import array

numbers = array.array('i', [1, 2, 3])

numbers.append(4)
print(numbers)  # array('i', [1, 2, 3, 4])

# extend() appends iterable to the end of the array
```

```
numbers.extend([5, 6, 7])
print(numbers)     # array('i', [1, 2, 3, 4, 5, 6, 7])
```

You can concatenate two arrays using + operator

```
import array

odd = array.array('i', [1, 3, 5])
even = array.array('i', [2, 4, 6])

integers = array.array('i')   # creating empty array of integer
integers = odd + even

print(integers)
```

How do you remove/delete elements?

To delete one or more items from an array, use the `del` statement as with lists.

```
import array

integer_array = array.array('i', [1, 2, 3, 3, 4])

del integer_array[2]  # removing third element
print(integer_array)  # Output: array('i', [1, 2, 3, 4])

del integer_array  # deleting entire array
print(integer_array)  # Error: array is not defined
```

We can use the `remove(val)` method to remove the given ite
`pop(index)` to remove an item at the given index. The `remove(val)` method removes the first element that is equal to `val` in the array.

> **Note:** An error is thrown if the index exceeds the size of the array or element is not found in the array.

```
import array

integer_array = array.array('i', [10, 11, 12, 12, 13])

integer_array.remove(12)
print(integer_array)   # array('i', [10, 11, 12, 13])

print(integer_array.pop(2))   # Output: 12
print(integer_array)   # array('i', [10, 11, 13])
```

In the next lesson, we'll be looking at the differences between arrays and lists in Python

Back
Lists

Next
Lists vs Arrays in Pyt'

Completed