How to create arrays?

As you might have guessed from the above example, we need to import array module to create arrays. For example:

```
1.
2. import array as arr
3. a = arr.array('d', [1.1, 3.5, 4.5])
4. print(a)
```

Here, we created an array of float type. The letter 'd' is a type code. This determines the type of the array during creation.
Commonly used type codes:

Code	C Type	Python Type	Min bytes
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	Py_UNICODE	Unicode	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	int	2
'l'	signed long	int	4
'L'	unsigned long	int	4
'f'	float	float	4
'd'	double	float	8

How to access array elements?

We use indices to access elements of an array:

```
1.
2. import array as arr
```

```
3. a = arr.array('i', [2, 4, 6, 8])
4.
5. print("First element:", a[0])
6. print("Second element:", a[1])
7. print("Second last element:", a[-1])
```

How to slice arrays?

We can access a range of items in an array by using the slicing operator :.

```
1.
2. import array as arr
3.
4. numbers_list = [2, 5, 62, 5, 42, 52, 48, 5]
5. numbers_array = arr.array('i', numbers_list)
6.
7. print(numbers_array[2:5]) # 3rd to 5th
8. print(numbers_array[:-5]) # beginning to 4th
9. print(numbers_array[5:])  # 6th to end
10. print(numbers_array[:])   # beginning to end
```

When you run the program, the output will be:

```
array('i', [62, 5, 42])
array('i', [2, 5, 62])
array('i', [52, 48, 5])
array('i', [2, 5, 62, 5, 42, 52, 48, 5])
```

How to change or add elements?

Arrays are mutable; their elements can be changed in a similar way like lists.

```
1.
2. import array as arr
3.
4. numbers = arr.array('i', [1, 2, 3, 5, 7, 10])
5.
6. # changing first element
7. numbers[0] = 0
8. print(numbers)     # Output: array('i', [0, 2, 3, 5, 7, 10])
9.
10. # changing 3rd to 5th element
```

```
numbers[2:5] = arr.array('i', [4, 6, 8])
print(numbers)     # Output: array('i', [0, 2, 4, 6, 8, 10])
```

We can add one item to a list using the append() method, or add several items using extend() method.

```

import array as arr

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

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

# extend() appends iterable to the end of the array
numbers.extend([5, 6, 7])
print(numbers)     # Output: array('i', [1, 2, 3, 4, 5, 6, 7])
```

We can concatenate two arrays using + operator.

```

import array as arr

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

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

print(numbers)
```

How to remove/delete elements?

We can delete one or more items from an array using Python's del statement.

```

import array as arr

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

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

```

```
9. del number # deleting entire array
10. print(number) # Error: array is not defined
```

We can use the remove() method to remove the given item, and pop() method to remove an item at the given index.

```
1.
2. import array as arr
3.
4. numbers = arr.array('i', [10, 11, 12, 12, 13])
5.
6. numbers.remove(12)
7. print(numbers)   # Output: array('i', [10, 11, 12, 13])
8.
9. print(numbers.pop(2))   # Output: 12
10. print(numbers)   # Output: array('i', [10, 11, 13])
```

When to use arrays?

Lists are much more flexible than arrays. They can store elements of different data types including string. Also, lists are faster than arrays. And, if you need to do mathematical computation on arrays and matrices, you are much better off using something like NumPy library.
Unless you don't really need arrays (array module may be needed to interface with C code), don't use them.