Python

Full stack Skills Bootcamp

Introducing Python Arrays

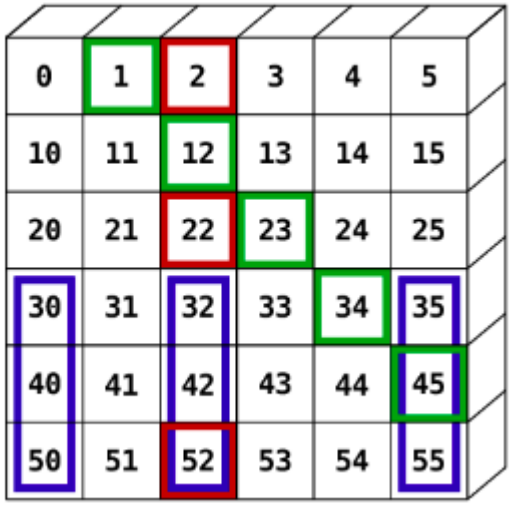
- What are arrays?

  Arrays, or lists in Python, are used to store multiple items in a single variable. They can hold a variety of data types and are mutable.

```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]
array([1, 12, 23, 34, 45])

>>> a[3:, [0,2,5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = np.array([1,0,1,0,0,1], dtype=bool)
>>> a[mask, 2]
array([2, 22, 52])
```

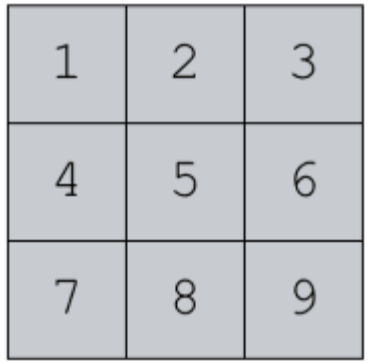


Creating Arrays

- Initialization

  Arrays are created by enclosing elements in square brackets.

```
Example: my_array = ["apple", "banana", "cherry"]
```



- Key Point:

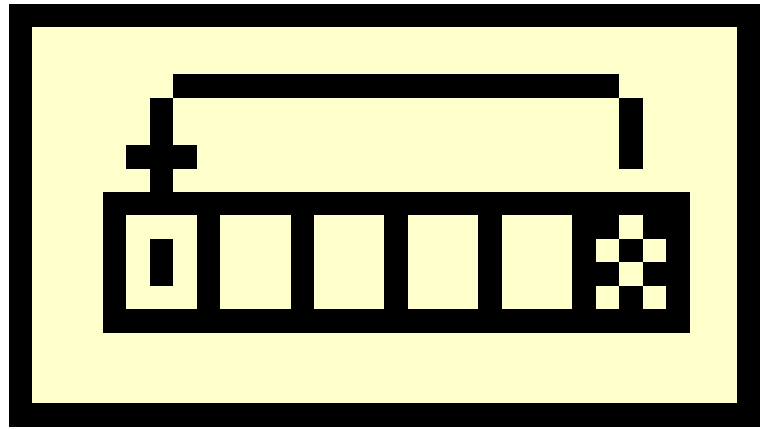
  Arrays can contain elements of different types (e.g., integers, strings).

Accessing and Modifying Array Elements

Using Indices

Elements are accessed using their index, starting from 0.

```python
first_element = my_array[0]  # Accesses "apple"
```



Updating Array Values

Elements can be modified by assigning a new value to a specific index.

```python
my_array[1] = "blueberry"  # Changes "banana" to "blueberry"
```

Slicing, and Adding Elements

- ## Creating Sub-arrays

  Use slicing to get a portion of the array.

Example:

python

```python
sub_array = my_array[1:3]  # Gets ["blueberry", "cherry"]
```

- ## Adding elements

  Use the append() method to add elements.

Example:

python

```python
sub_array = my_array[1:3]  # Gets ["blueberry", "cherry"]
```

Removing Elements

Removing Elements

Here are some other ways to remove elements from a list in Python.

- remove()
- pop()
- del
- list comprehension
- clear()

Method	Description	Example Code	Output
`remove(item)`	Removes the first occurrence of the specified element.	`my_array.remove("cherry")`	`["apple", "banana", "apple", "date", "fig", "grape"]`
`pop(index)`	Removes and returns the element at the specified index (default: last).	`removed_element = my_array.pop(2)`	List: `["apple", "banana", "date", "fig", "grape"]` Removed Element: `"date"`
`del list[index]`	Deletes the element at the specified index.	`del my_array[1]`	`["apple", "date", "fig", "grape"]`
`del list[start:end]`	Deletes a range of elements from the list.	`del my_array[1:3]`	`["apple", "grape"]`
List comprehension	Creates a new list without the specified element(s).	`my_array = [item for item in my_array if item != "apple"]`	`["banana", "cherry", "date", "fig", "grape"]`
`clear()`	Removes all elements from the list.	`my_array.clear()`	`[]`

Conclusion

- Arrays (lists) are powerful tools for storing and manipulating data.
- Understanding array methods enhances code efficiency and readability.