Microcontrollers & Applications

Lecture 2.3: Arrays & String & Lists

Arduino & CircuitPython Structure

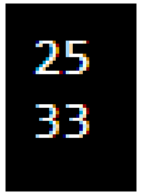
```
// Library inclusion declarations
// Global variables and constants definitions

void setup() {
        // Whatever you code here,
        // it runs first but once only
}

void loop() {
        // Whatever you code here,
        // it runs until electricty is down
}
```

```
# Library inclusion declarations
# Global variables and constants definitions

# Whatever you code here,
# it runs first but once only

while (True):
        # Whatever you code here,
        # it runs until electricty is down
```

C Arrays (1)

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To create an array, define the data type (like int) and specify the name of the array followed by square brackets [ ].
- To insert values to it, use a comma-separated list, inside curly braces.
- To access an array element, refer to its index number inside square brackets [ ].
- Array indexes start with 0.

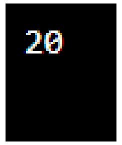
```
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  printf("%d\n", myNumbers[0]);

  myNumbers[0] = 33;
  printf("%d\n", myNumbers[0]);

  return 0;
}
```



```
25
33
```

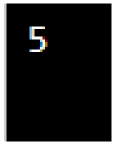
C Arrays (2)

```
#include <stdio.h>

int main() {
  int myNumbers[] = {10, 25, 50, 75, 100};
  printf("%lu", sizeof(myNumbers));

  return 0;
}
```



```
20
```

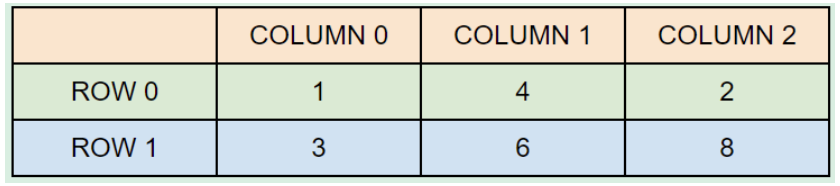
```
#include <stdio.h>

int main() {
  int myNumbers[] = {10, 25, 50, 75, 100};
  int length = sizeof(myNumbers) /
sizeof(myNumbers[0]);

  printf("%d", length);
  return 0;
}
```



```
5
```

C Arrays (3)

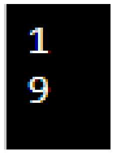
```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
```



	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

```
int LED_PINS[] = {6, 7, 9, 10, 11, 5, 4}; // Arduino Example
```

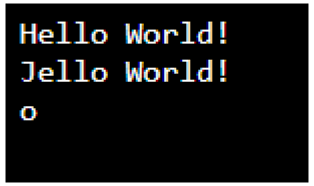
```
#include <stdio.h>

int main() {
  int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

  printf("%d\n", matrix[0][0]);

  matrix[0][0] = 9;
  printf("%d\n", matrix[0][0]);

  return 0;
}
```



```
1
9
```

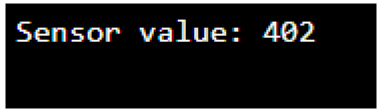
C Character Array versus String

```
#include <stdio.h>

int main() {
  char greetings[] = "Hello World!";
  printf("%s\n ", greetings);
  greetings[0] = 'J';
  printf("%s\n ", greetings);
  printf("%c\n", greetings[4]);

  return 0;
}
```



```
Hello World!
Jello World!
o
```

```
// in Arduino
int sensorValue = 402;
String stringOne = "Sensor value: ";
String stringThree = stringOne + sensorValue;
Serial.println(stringThree);
```



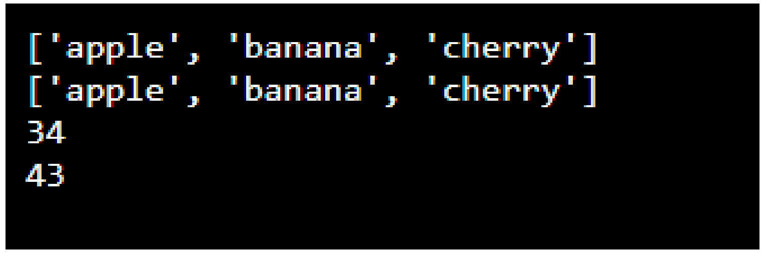
```
Sensor value: 402
```

Python Arrays (1)

Python Collections (Arrays)

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered, unchangeable, and unindexed. No duplicate members.
- Dictionary is a collection which is ordered and changeable. No duplicate members.

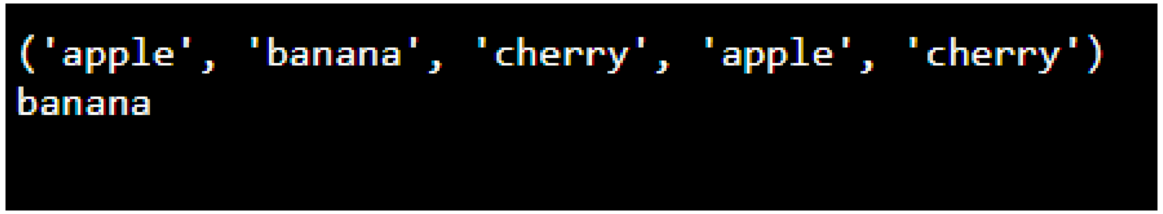
```python
list1 = ["apple", "banana", "cherry"]
list2 = list(("apple", "banana", "cherry"))
list3 = [1, 5, 7, 9, 3]
list4 = [True, False, False]
list5 = ["abc", 34, True, 40, "male"]


print(list1)
print(list2)
print(list5[1])
list5[1] = 43
print(list5[1])
```



```
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry']
34
43
```

Python Arrays (2)

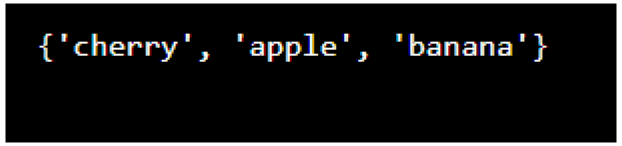
```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
print(thistuple[1])

thistuple[1] = "orange"  #this gives error
```



```
('apple', 'banana', 'cherry', 'apple', 'cherry')
banana
```

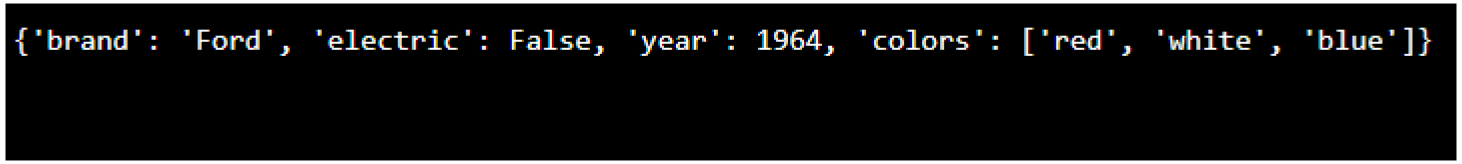
Python Arrays (3)

```
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
print(thisset[1])    #this gives error
```



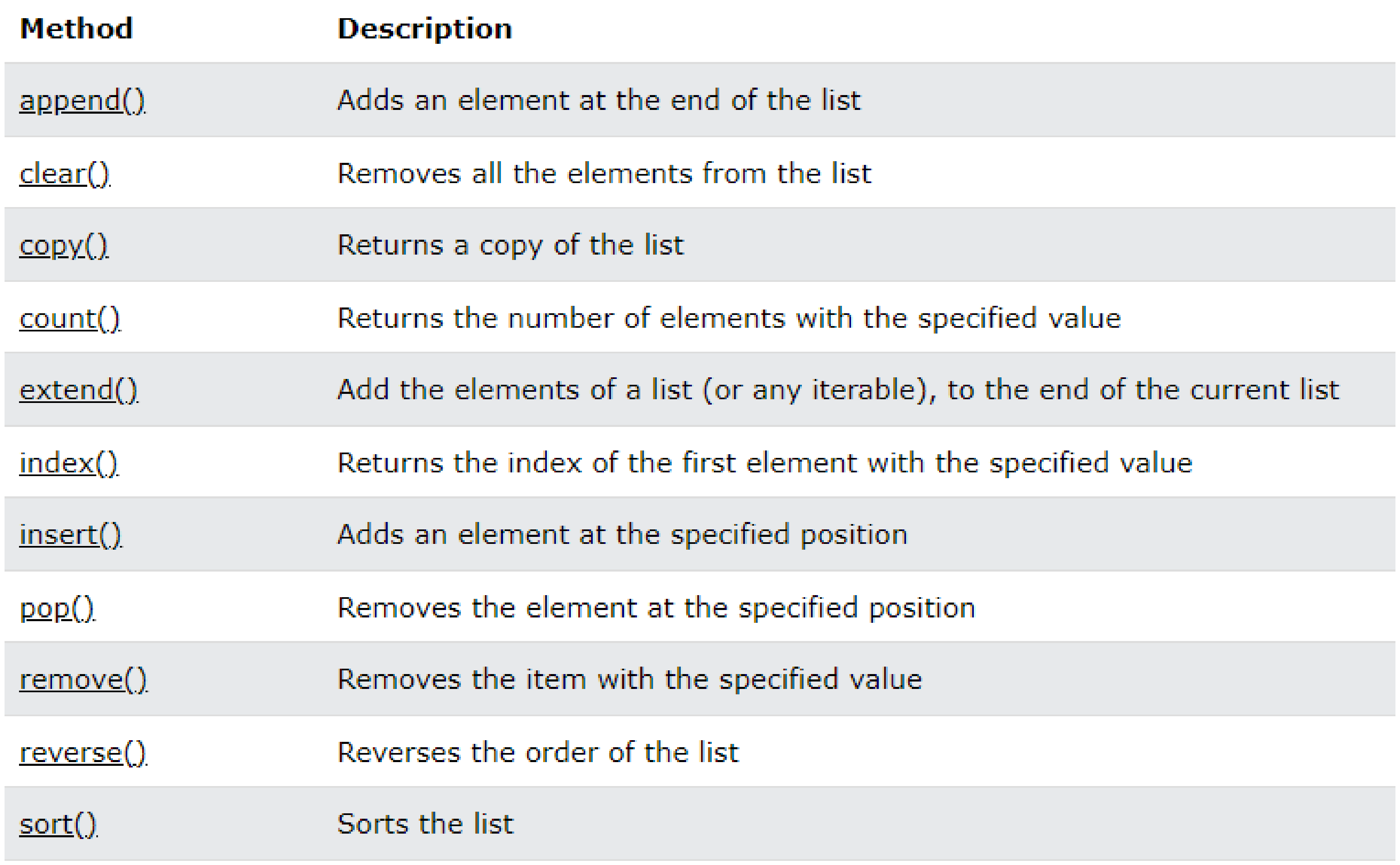
```
{'cherry', 'apple', 'banana'}
```

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}

print(thisdict)
```



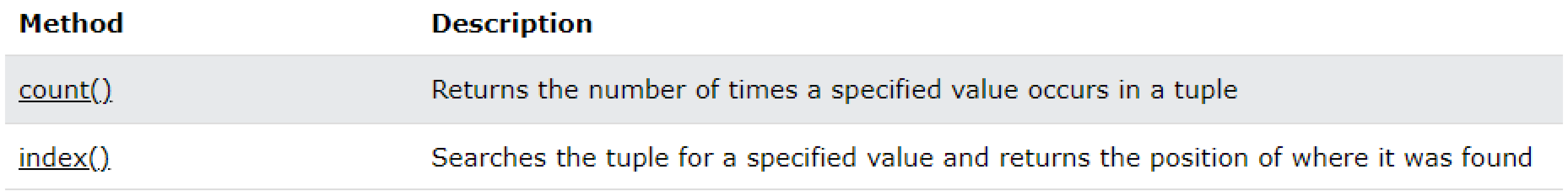
```
{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}
```

Python List Methods



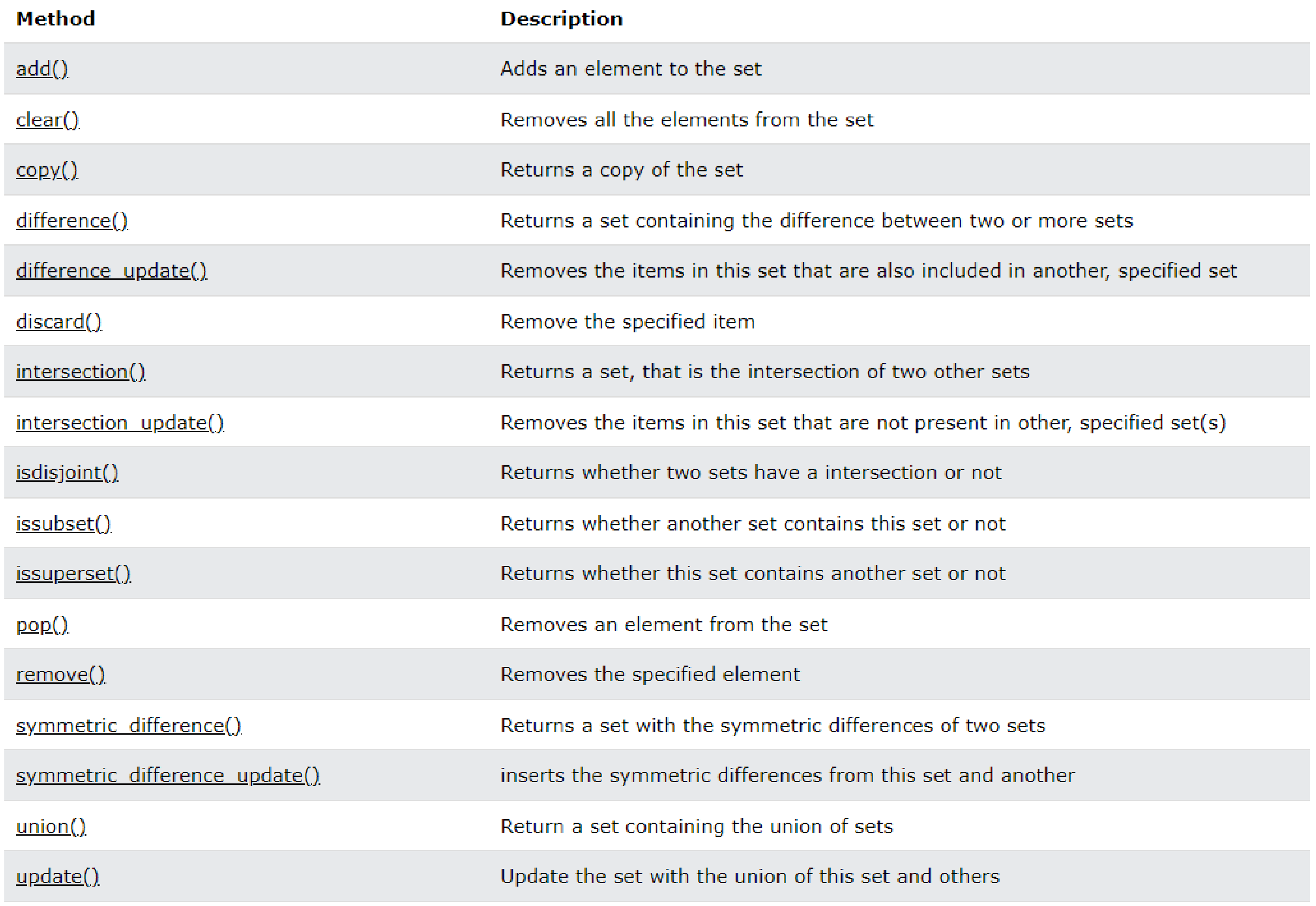
Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	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 item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Tuple Methods



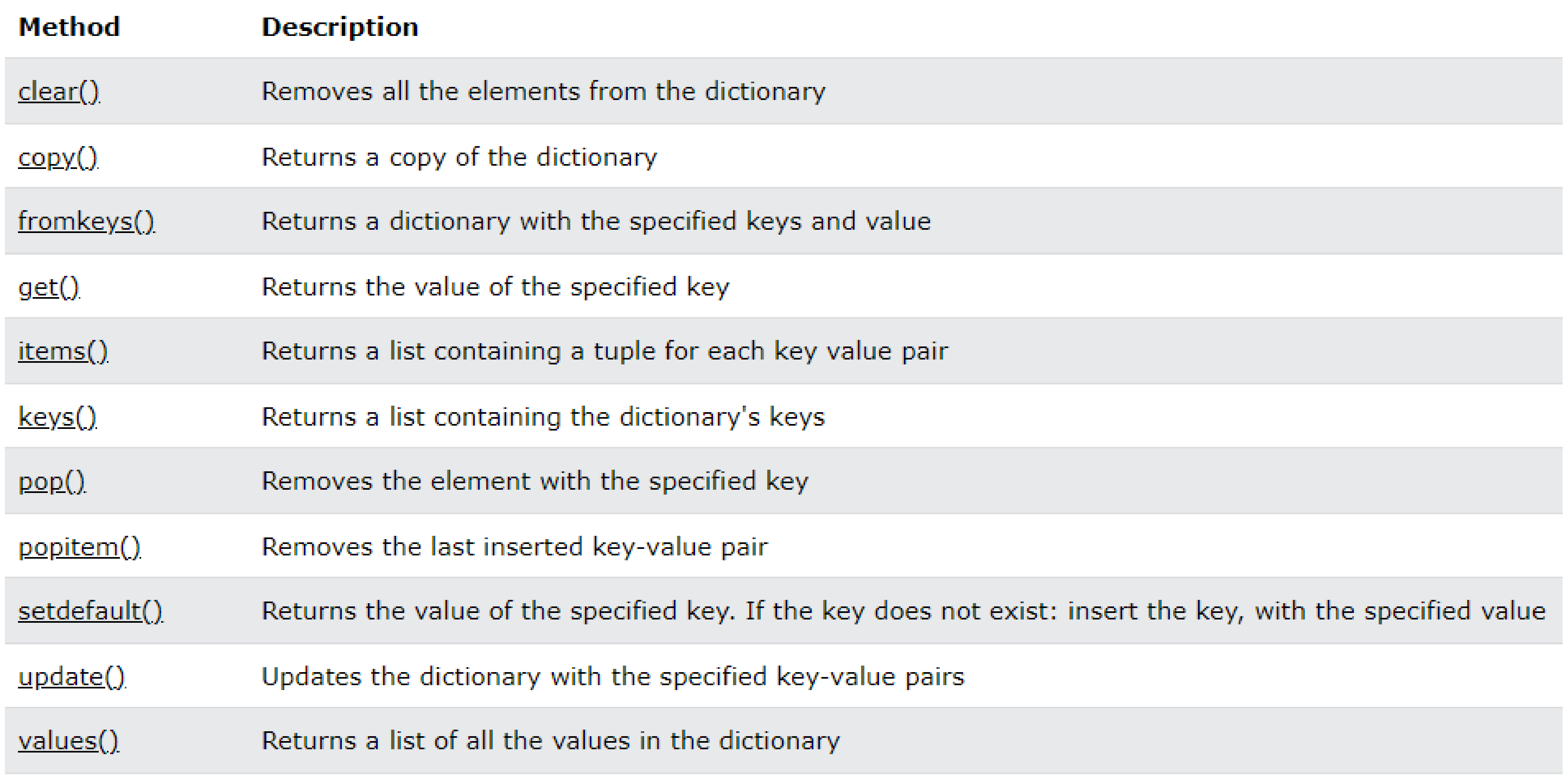
Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Set Methods



Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Python Dictionary Methods



Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Thanks for listening ☺

YALÇIN İŞLER

Assoc. Prof.