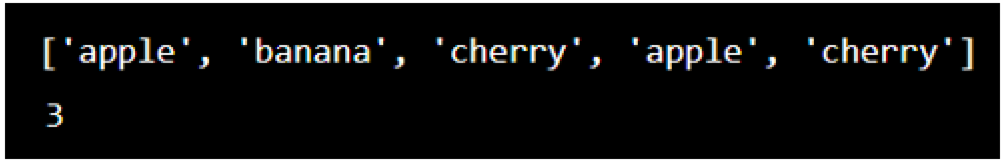
Introduction to Python Datatypes

21ES601 Embedded System Programming

List

- Lists are used to store multiple items in a single variable
- List items are ordered, changeable, and allow duplicate values
  - If a new item is added to a list, the new items will be placed at the end of the list

```
thislist =
["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
print(len(thislist))
```

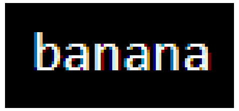


```
['apple', 'banana', 'cherry', 'apple', 'cherry']
3
```

List – Indexing

List items are indexed, the first item has index [0], the second item has index [1] etc.,

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

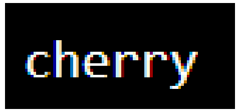


```
banana
```

Negative indexing means start from the end.

-1 refers to the last item, -2 refers to the second last item etc.

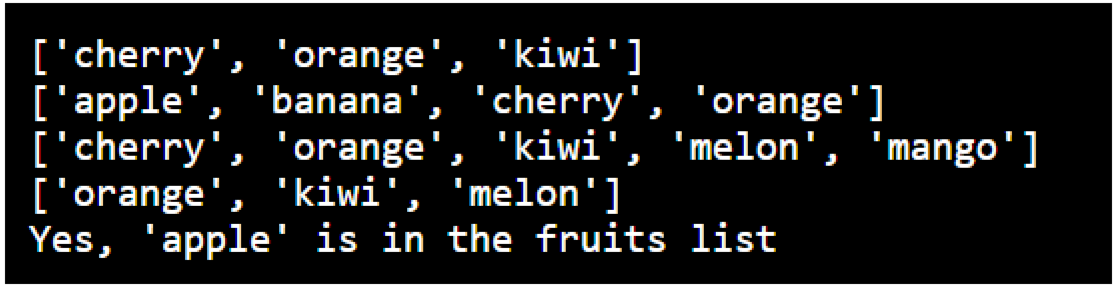
```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```



```
cherry
```

List – Indexing

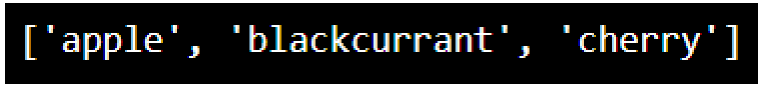
```
thislist =["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
print(thislist[:4])
print(thislist[2:])
print(thislist[-4:-1])
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```



```
['cherry', 'orange', 'kiwi']
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
['orange', 'kiwi', 'melon']
Yes, 'apple' is in the fruits list
```

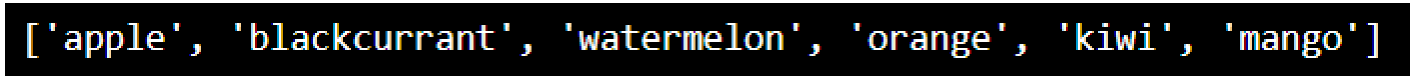
List – Update

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



```
['apple', 'blackcurrant', 'cherry']
```

```
thislist =
["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```



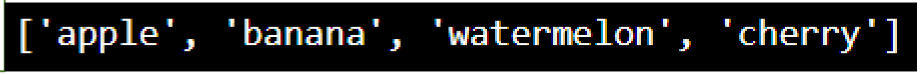
```
['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango']
```

```
thislist = ["apple", "banana", "cherry"]
thislist[1:3] = ["watermelon"]
print(thislist)
```

List – Update

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

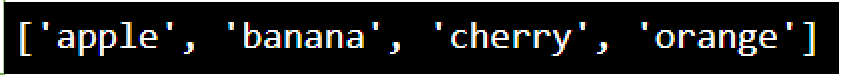
Insertion in the middle



```
['apple', 'banana', 'watermelon', 'cherry']
```

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

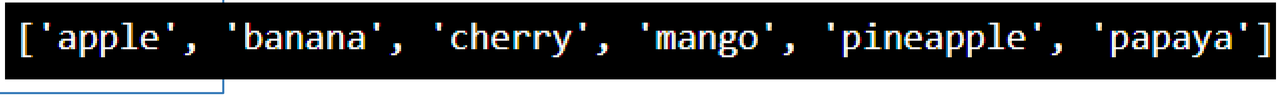
Insertion at the end



```
['apple', 'banana', 'cherry', 'orange']
```

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

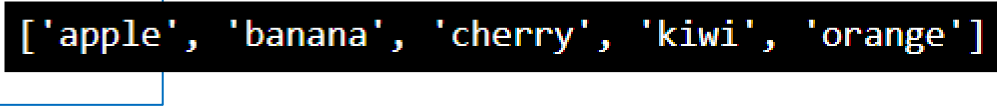
Join lists



```
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

```
thislist = ["apple", "banana", "cherry"]
thistuple = ("kiwi", "orange")
thislist.extend(thistuple)
print(thislist)
```

Join iterables



```
['apple', 'banana', 'cherry', 'kiwi', 'orange']
```

List – Item removal

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Removes specified Item

Removes item of specified index

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

Removes Last item

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Deletes entire list

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Clears the content of the list

List – Loop

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1
```

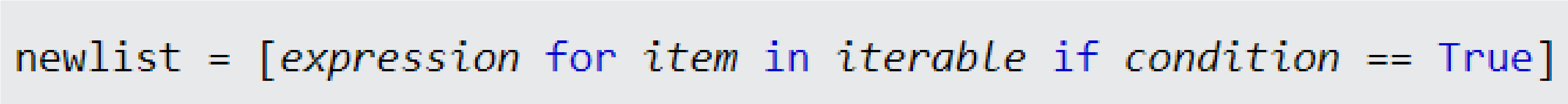
```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

List - Comprehension

Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name

```
fruits =
["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

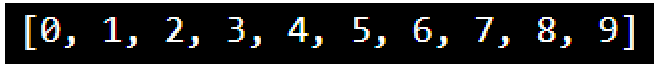
```
fruits =
["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```



```
newlist = [expression for item in iterable if condition == True]
```

List - Comprehension

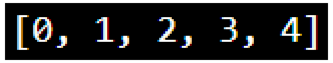
```
newlist = [x for x in range(10)]
```



```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Accept only numbers lower than 5

```
newlist = [x for x in range(10) if x < 5]
```

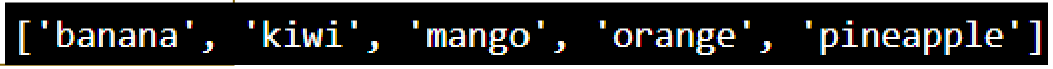


```
[0, 1, 2, 3, 4]
```

List – Sorting

Alphabet Sort

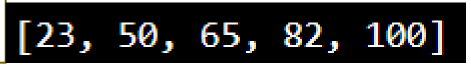
```
thislist =
["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```



```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Number Sort - Ascending

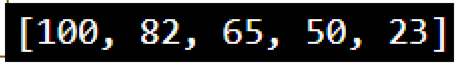
```
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```



```
[23, 50, 65, 82, 100]
```

Number Sort - Descending

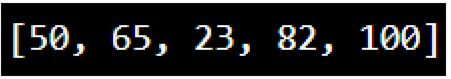
```
thislist = [100, 50, 65, 82, 23]
thislist.sort(reverse = True)
print(thislist)
```



```
[100, 82, 65, 50, 23]
```

Custom Sort

```
def myfunc(n):
  return abs(n - 50)

thislist = [100, 50, 65, 82, 23]
thislist.sort(key = myfunc)
print(thislist)
```



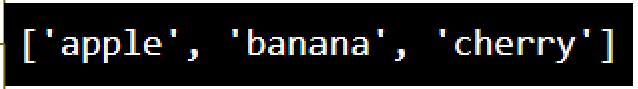
```
[50, 65, 23, 82, 100]
```

List – Copy, Extend

list2 = list1 is not a copy, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

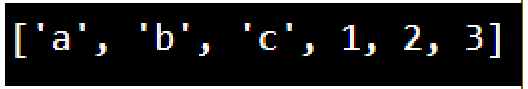


```
['apple', 'banana', 'cherry']
```

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
```

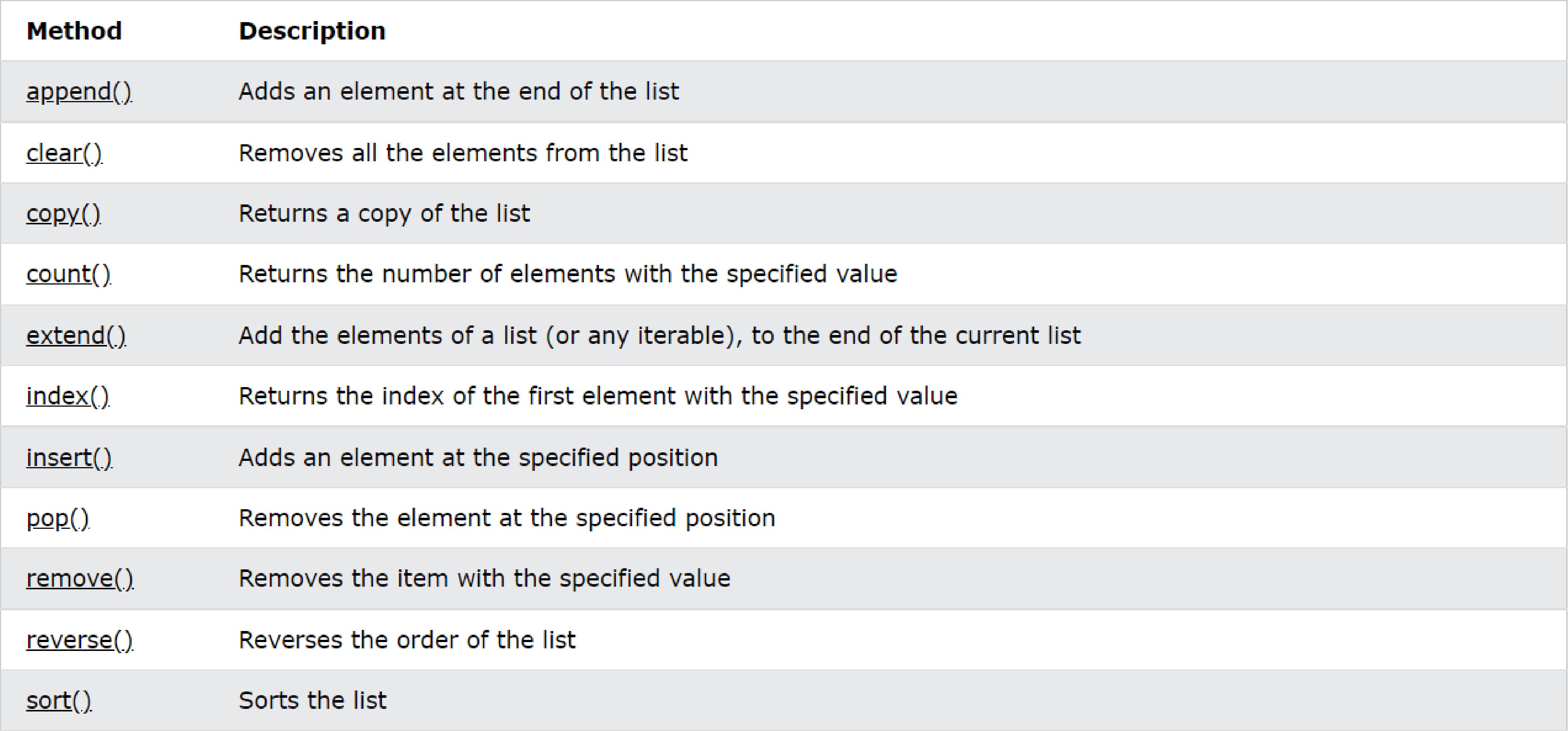
```
for x in list2:
  list1.append(x)
print(list1)
```

```
list1.extend(list2)
print(list1)
```



```
['a', 'b', 'c', 1, 2, 3]
```

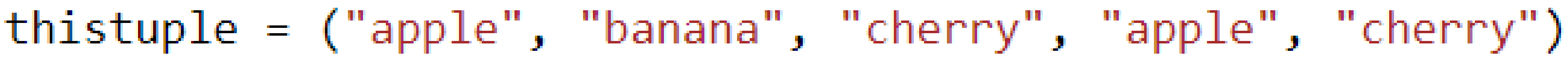
List – Methods



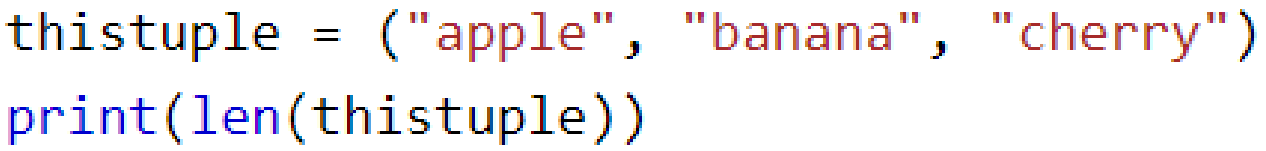
Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()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

Tuples

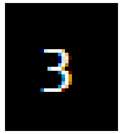
- Tuples are used to store multiple items in a single variable
- A collection which is ordered, unchangeable and allow duplicate values
- Tuples are written with round brackets



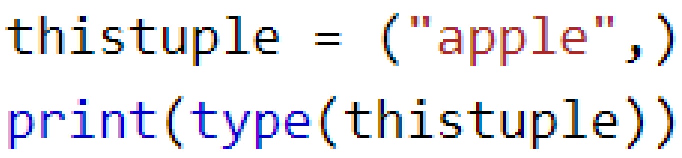
```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
```



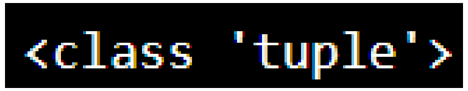
```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```



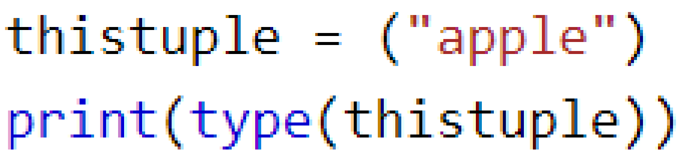
```
3
```



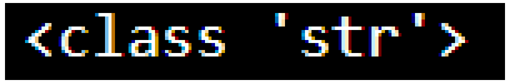
```
thistuple = ("apple",)
print(type(thistuple))
```



```
<class 'tuple'>
```



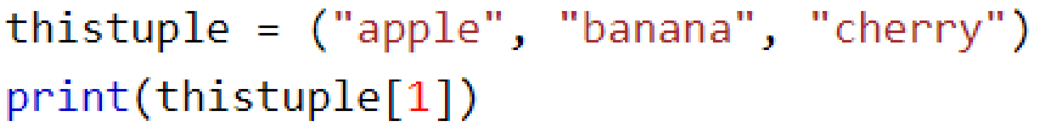
```
thistuple = ("apple")
print(type(thistuple))
```



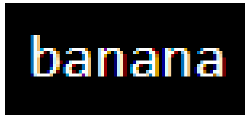
```
<class 'str'>
```

Tuples - Indexing

Items are indexed, the first item has index [0], the second item has index [1] etc



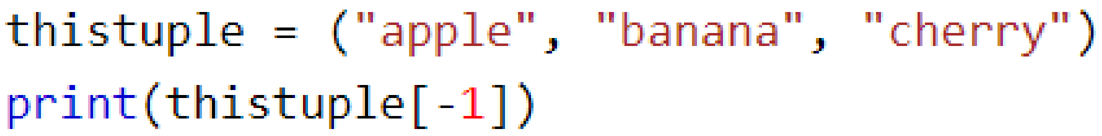
```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```



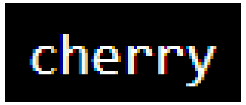
```
banana
```

Negative indexing means start from the end.

-1 refers to the last item, -2 refers to the second last item etc.



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



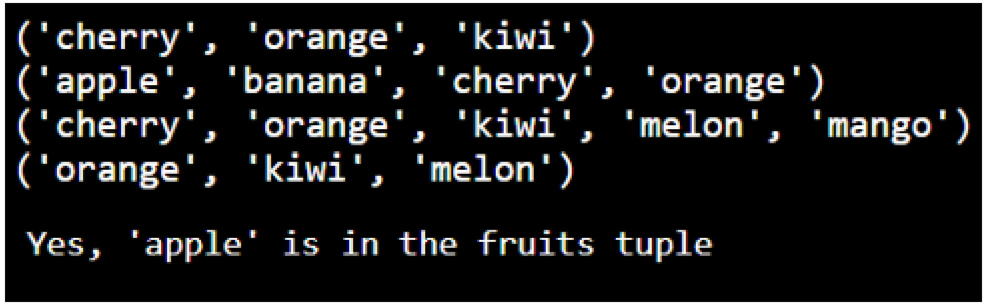
```
cherry
```

Tuples - Access

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
print(thistuple[:4])
print(thistuple[2:])
print(thistuple[-4:-1])
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

Tuples - Access

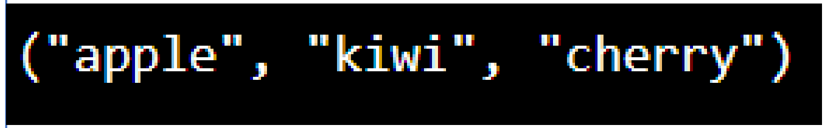
```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
print(thistuple[:4])
print(thistuple[2:])
print(thistuple[-4:-1])
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```



```
('cherry', 'orange', 'kiwi')
('apple', 'banana', 'cherry', 'orange')
('cherry', 'orange', 'kiwi', 'melon', 'mango')
('orange', 'kiwi', 'melon')

 Yes, 'apple' is in the fruits tuple
```

Tuples - Update

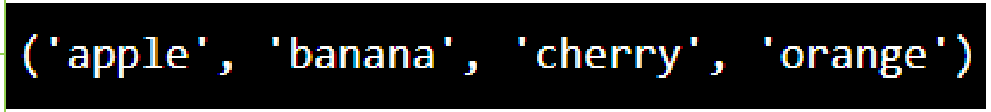
```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```



```
("apple", "kiwi", "cherry")
```

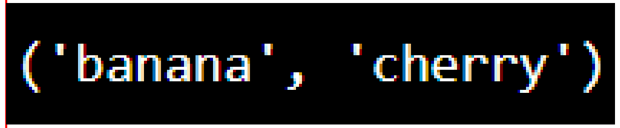
```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```



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

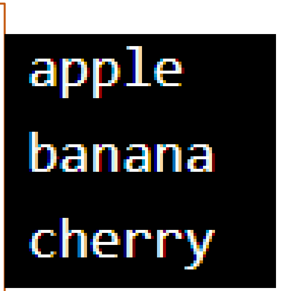
```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```



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

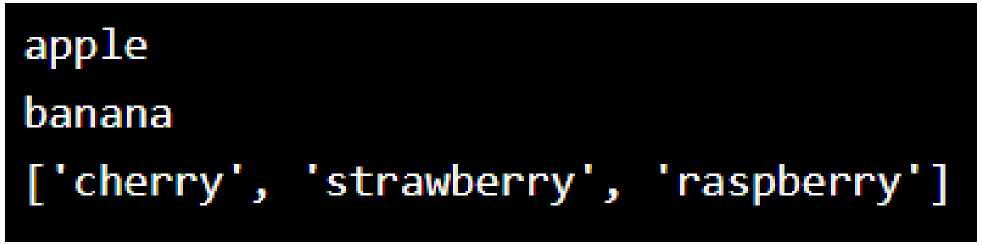
Tuple - Unpack

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)
print(yellow)
print(red)
```



```
apple
banana
cherry
```

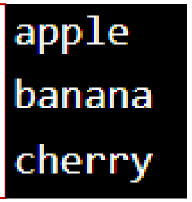
```
fruits =
("apple", "banana", "cherry", "strawberry", "ra
spberry")

(green, yellow, *red) = fruits

print(green)
print(yellow)
print(red)
```



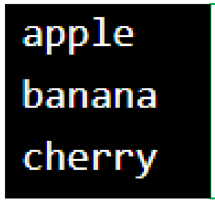
```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

Tuples – Loop

```python
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```



```
apple
banana
cherry
```



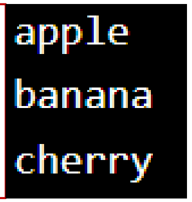
```
apple
banana
cherry
```

```python
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
  print(thistuple[i])
```

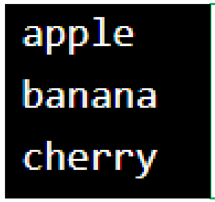
While loop?

Tuples – Loop

```python
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```



```
apple
banana
cherry
```



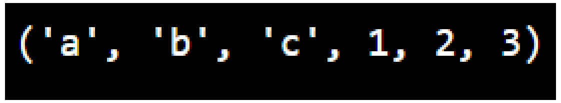
```
apple
banana
cherry
```

```python
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
  print(thistuple[i])
```

```python
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
  print(thistuple[i])
  i = i + 1
```

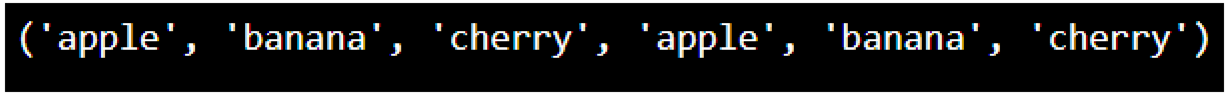
Tuples - Join

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```



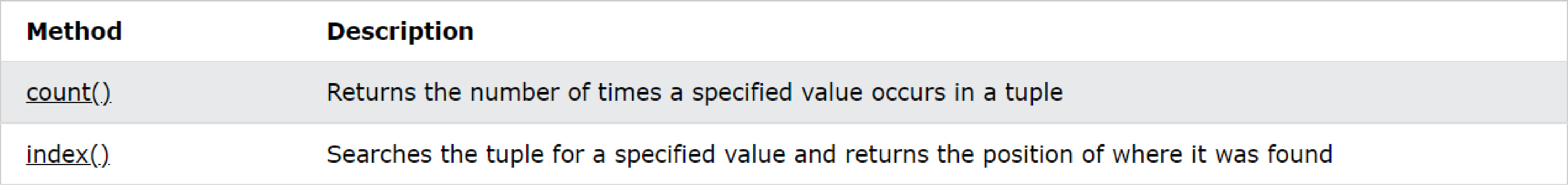
```
('a', 'b', 'c', 1, 2, 3)
```

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2

print(mytuple)
```



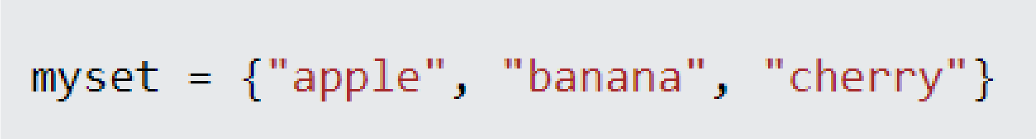
```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

Tuples – 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

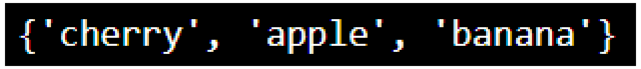
Set



```
myset = {"apple", "banana", "cherry"}
```

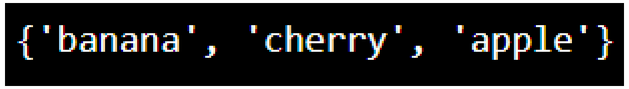
- Sets are used to store multiple items in a single variable
- Collection which is unordered, unchangeable and unindexed
  - But items can be added and removed
- Duplicate entries are not allowed

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```



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

```
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
```



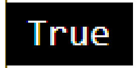
```
{'banana', 'cherry', 'apple'}
```

Set – Access

- Items are not indexed
- Can be accessed through for loop

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

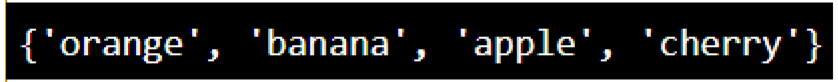
```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```



```
True
```

Set – Update

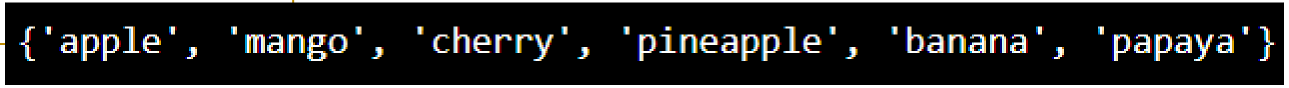
```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```



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

Add items from another set into the current set

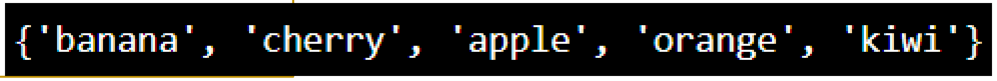
```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```



```
{'apple', 'mango', 'cherry', 'pineapple', 'banana', 'papaya'}
```

The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.)

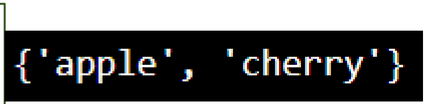
```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)
```



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

Set – Remove

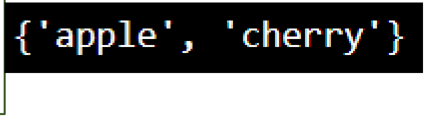
```python
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```



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

Remove an item in a set

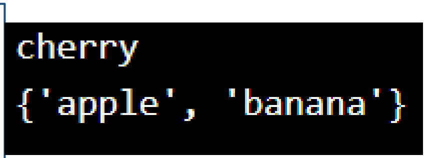
```python
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```



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

If the item to remove does not exist, remove() will raise an error.
If the item to remove does not exist, discard() will NOT raise an error.

```python
thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x)
print(thisset)
```



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

Remove last item in a set

Pop() method will remove the last item. Remember that sets are unordered, so the item that gets removed is unknown

Set – Remove

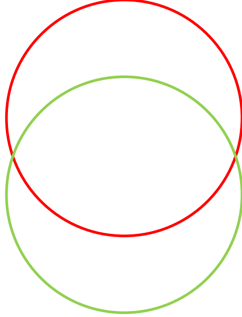
```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

Removes all items in a set

```
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)
```

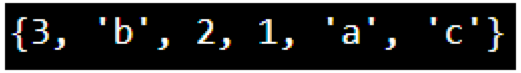
Delete a set completely

Set – Join



```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

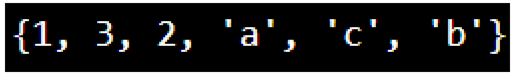
Returns a new set with all items from both sets



```
{3, 'b', 2, 1, 'a', 'c'}
```

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

Inserts the items in set2 into set1

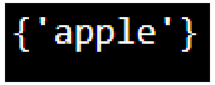


```
{1, 3, 2, 'a', 'c', 'b'}
```

union() and update() will exclude any duplicate items

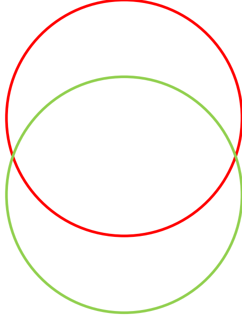
```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)
```

Keep only the items that are present in both sets



```
{'apple'}
```

Set – Join

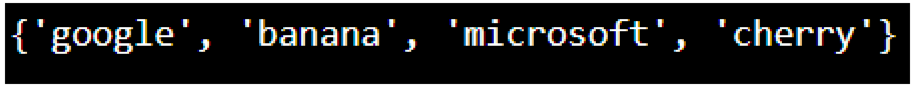


```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y)
print(z)
```

Return a set that contains the items that exist in both set x, and set y

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y)
print(x)
```

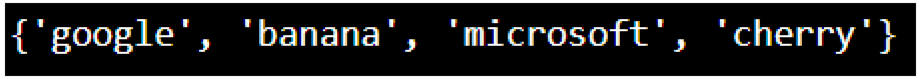
Keep the items that are not present in both sets



```
{'google', 'banana', 'microsoft', 'cherry'}
```

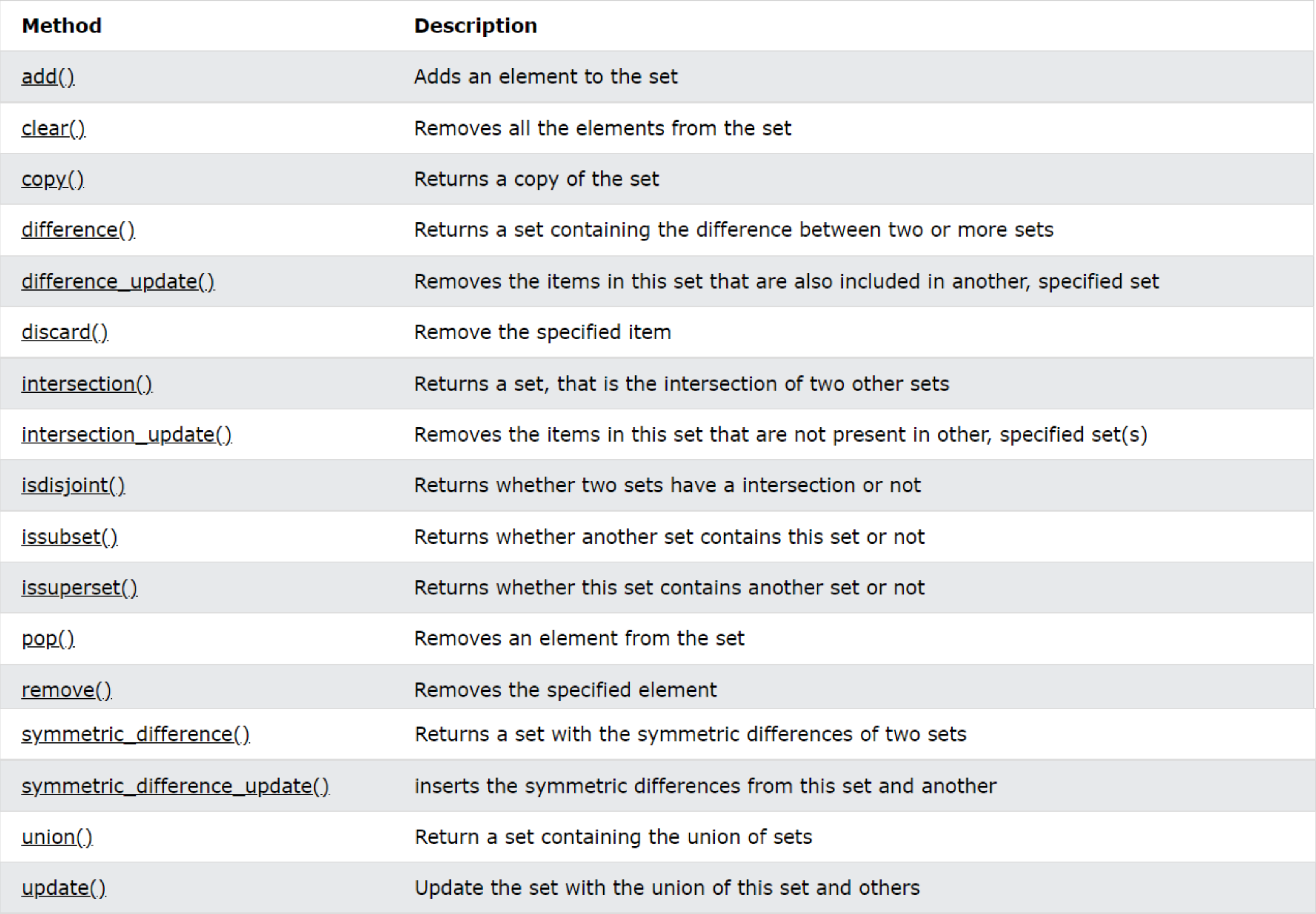
```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.symmetric_difference(y)
print(z)
```

Return a set that contains all items from both sets, except items that are present in both



```
{'google', 'banana', 'microsoft', 'cherry'}
```

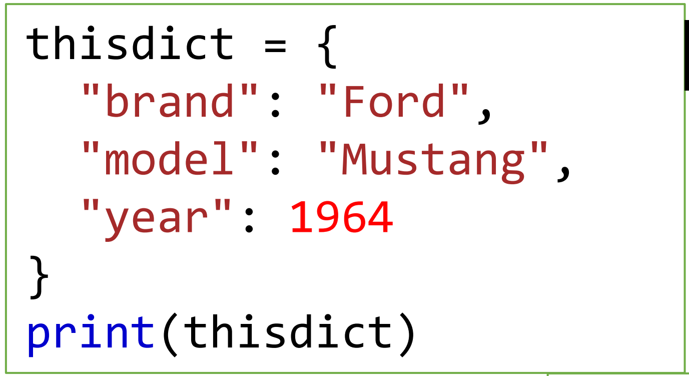
Set – Methods



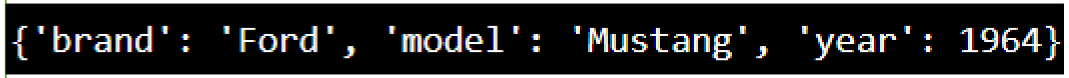
Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified 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

Dictionary

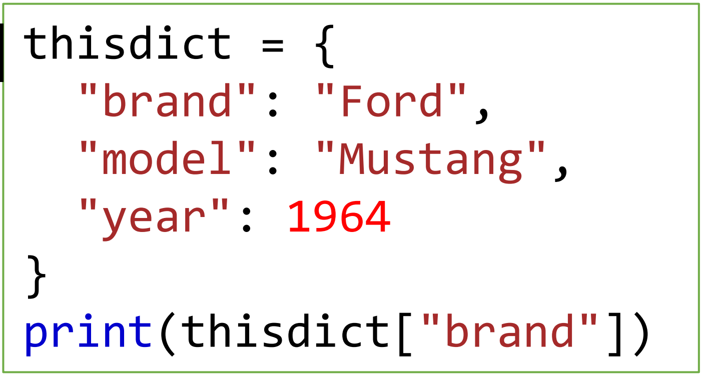
- Dictionaries are used to store data values in key:value pairs
- A dictionary is a collection which is ordered, changeable and do not allow duplicates



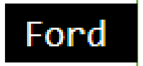
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```



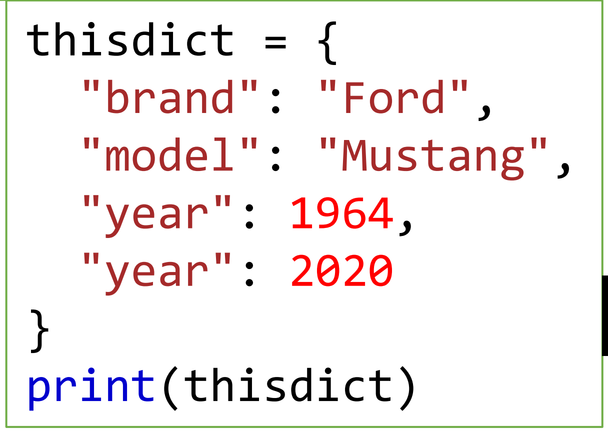
```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```



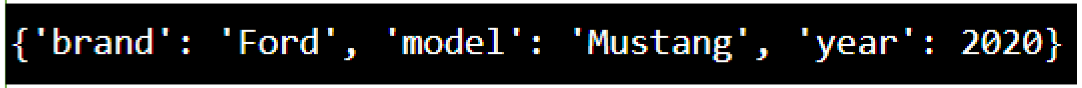
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```



```
Ford
```



```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```



```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Dictionary

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

print(len(thisdict))

print(type(thisdict))
```

The values in dictionary items can be of any data type

Print the number of items in the dictionary

```
thisdict = dict(name = "John", age = 36,
country = "Norway")
print(thisdict)
```

Method to make a dictionary

Dictionary – Access

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
```

Items of a dictionary can be accessed by referring to its key name, inside square brackets

```
y = thisdict.get("model")
```

Method to access a key

```
z = thisdict.keys()
```

Method to list all keys in a dictionary

```
u = thisdict.values()
```

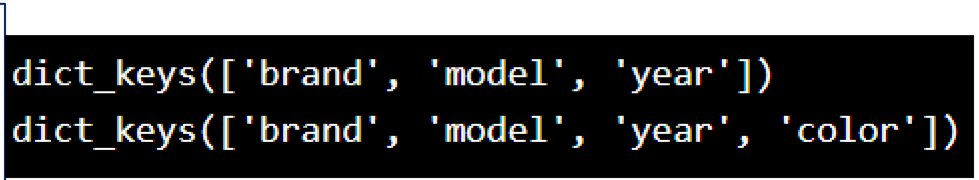
Method to list all values in a dictionary

```
v = thisdict.items()
```

Method to return each item in a dictionary, as tuples in a list

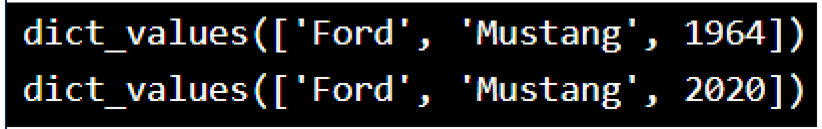
Dictionary – Access

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.keys()
print(x) #before the change
car["color"] = "white"
print(x) #after the change
```



```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
```

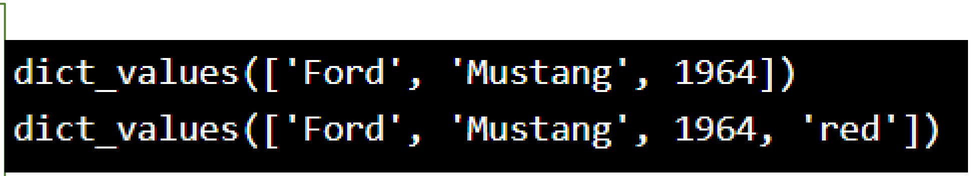
```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.values()
print(x) #before the change
car["year"] = 2020
print(x) #after the change
```



```
dict_values(['Ford', 'Mustang', 1964])
dict_values(['Ford', 'Mustang', 2020])
```

Dictionary – Update

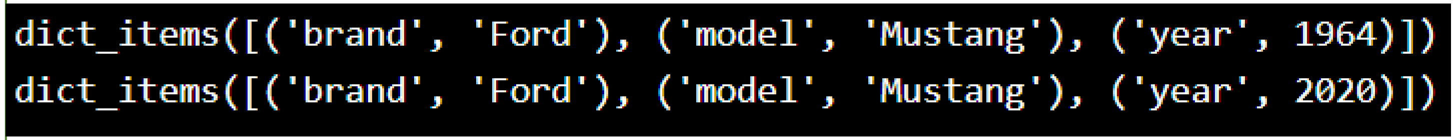
```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.values()
print(x) #before the change
car["color"] = "red"
print(x) #after the change
```



```
dict_values(['Ford', 'Mustang', 1964])
dict_values(['Ford', 'Mustang', 1964, 'red'])
```

Add key:value pair

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.items()
print(x) #before the change
car["year"] = 2020
print(x) #after the change
```

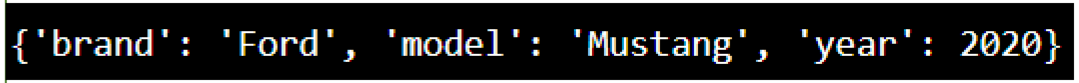


```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2020)])
```

Update key:value pair

Dictionary – Update

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
```

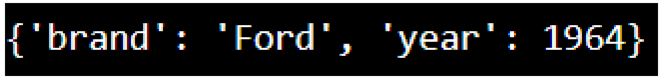


```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Update key:value pair

Dictionary - Remove

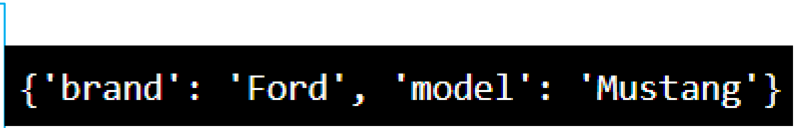
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```



```
{'brand': 'Ford', 'year': 1964}
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

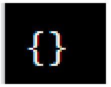


```
{'brand': 'Ford', 'model': 'Mustang'}
```

Dictionary - Remove

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```



```
{}
```

Dictionary and Loop

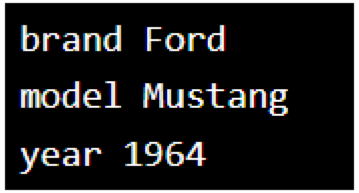
```
for x in thisdict:
  print(x)

for x in thisdict.keys():
  print(x)
```

Print all key names in the dictionary, one by one

```
for x in thisdict:
  print(thisdict[x])

for x in thisdict.values():
  print(x)
```

Print all *values* in the dictionary, one by one

```
for x, y in thisdict.items():
  print(x, y)
```



```
brand Ford
model Mustang
year 1964
```

Dictionary – Copy

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

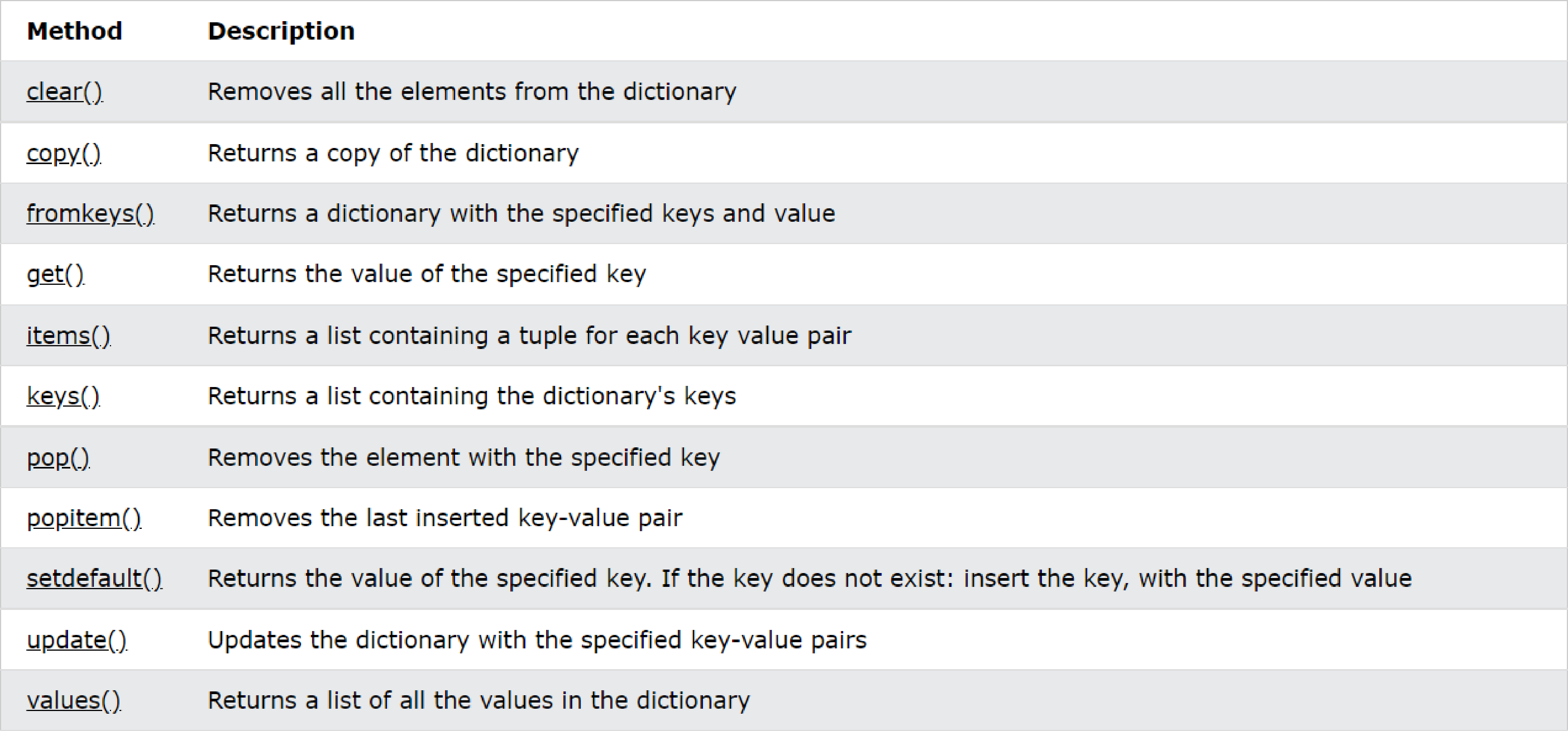
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

Dictionary – Nested

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
```

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

Comparison

There are four collection data types in the Python programming language:

- 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
  - Set items are unchangeable, but one can remove items and add new items
- Dictionary is a collection which is ordered and changeable. No duplicate members

Mutable and Immutable types

- A mutable object is an object whose state can be modified after it is defined
- Eg: List, Set, Dictionary
- An immutable object is an object whose state cannot be altered after it is initially defined
- Eg: int, float, bool, string, unicode, tuple

References

- https://www.w3schools.com/python/default.asp