Python Iterators

[❮ Previous](https://www.w3schools.com/python/python_inheritance.asp)[Next ❯](https://www.w3schools.com/python/python_polymorphism.asp)

Python Iterators

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods \_\_iter\_\_() and \_\_next\_\_().

Iterator vs Iterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable *containers* which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Return an iterator from a tuple, and print each value:

mytuple = ("apple", "banana", "cherry")  
myit = iter(mytuple)  
  
print(next(myit))  
print(next(myit))  
print(next(myit))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator)

Even strings are iterable objects, and can return an iterator:

Example

Strings are also iterable objects, containing a sequence of characters:

mystr = "banana"  
myit = iter(mystr)  
  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator2)

Looping Through an Iterator

We can also use a for loop to iterate through an iterable object:

Example

Iterate the values of a tuple:

mytuple = ("apple", "banana", "cherry")  
  
for x in mytuple:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_loop)

Example

Iterate the characters of a string:

mystr = "banana"  
  
for x in mystr:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_loop2" \t "_blank)

The for loop actually creates an iterator object and executes the next() method for each loop.

ADVERTISEMENT

Create an Iterator

To create an object/class as an iterator you have to implement the methods \_\_iter\_\_() and \_\_next\_\_() to your object.

As you have learned in the [Python Classes/Objects](https://www.w3schools.com/python/python_classes.asp) chapter, all classes have a function called \_\_init\_\_(), which allows you to do some initializing when the object is being created.

The \_\_iter\_\_() method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.

The \_\_next\_\_() method also allows you to do operations, and must return the next item in the sequence.

Example

Create an iterator that returns numbers, starting with 1, and each sequence will increase by one (returning 1,2,3,4,5 etc.):

class MyNumbers:  
  def \_\_iter\_\_(self):  
    self.a = 1  
    return self  
  
  def \_\_next\_\_(self):  
    x = self.a  
    self.a += 1  
    return x  
  
myclass = MyNumbers()  
myiter = iter(myclass)  
  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_create)

StopIteration

The example above would continue forever if you had enough next() statements, or if it was used in a for loop.

To prevent the iteration from going on forever, we can use the StopIteration statement.

In the \_\_next\_\_() method, we can add a terminating condition to raise an error if the iteration is done a specified number of times:

Example

Stop after 20 iterations:

class MyNumbers:  
  def \_\_iter\_\_(self):  
    self.a = 1  
    return self  
  
  def \_\_next\_\_(self):  
    if self.a <= 20:  
      x = self.a  
      self.a += 1  
      return x  
    else:  
      raise StopIteration  
  
myclass = MyNumbers()  
myiter = iter(myclass)  
  
for x in myiter:  
  print(x)

Python Polymorphism

[❮ Previous](https://www.w3schools.com/python/python_iterators.asp)[Next ❯](https://www.w3schools.com/python/python_scope.asp)

The word "polymorphism" means "many forms", and in programming it refers to methods/functions/operators with the same name that can be executed on many objects or classes.

Function Polymorphism

An example of a Python function that can be used on different objects is the len() function.

String

For strings len() returns the number of characters:

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x = "Hello World!"  
  
print(len(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_1)

Tuple

For tuples len() returns the number of items in the tuple:

Example

mytuple = ("apple", "banana", "cherry")  
  
print(len(mytuple))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_2)

Dictionary

For dictionaries len() returns the number of key/value pairs in the dictionary:

Example

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_3)

ADVERTISEMENT

Class Polymorphism

Polymorphism is often used in Class methods, where we can have multiple classes with the same method name.

For example, say we have three classes: Car, Boat, and Plane, and they all have a method called move():

Example

Different classes with the same method:

class Car:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Drive!")  
  
class Boat:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Sail!")  
  
class Plane:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Fly!")  
  
car1 = Car("Ford", "Mustang")       #Create a Car class  
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat class  
plane1 = Plane("Boeing", "747")     #Create a Plane class  
  
for x in (car1, boat1, plane1):  
  x.move()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_4)

Look at the for loop at the end. Because of polymorphism we can execute the same method for all three classes.

Inheritance Class Polymorphism

What about classes with child classes with the same name? Can we use polymorphism there?

Yes. If we use the example above and make a parent class called Vehicle, and make Car, Boat, Plane child classes of Vehicle, the child classes inherits the Vehicle methods, but can override them:

Example

Create a class called Vehicle and make Car, Boat, Plane child classes of Vehicle:

class Vehicle:  
  def \_\_init\_\_(self, brand, model):  
    self.brand = brand  
    self.model = model  
  
  def move(self):  
    print("Move!")  
  
class Car(Vehicle):  
  pass  
  
class Boat(Vehicle):  
  def move(self):  
    print("Sail!")  
  
class Plane(Vehicle):  
  def move(self):  
    print("Fly!")  
  
car1 = Car("Ford", "Mustang") #Create a Car object  
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object  
plane1 = Plane("Boeing", "747") #Create a Plane object  
  
for x in (car1, boat1, plane1):  
  print(x.brand)  
  print(x.model)  
  x.move()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_poly_5)

Child classes inherits the properties and methods from the parent class.

In the example above you can see that the Car class is empty, but it inherits brand, model, and move() from Vehicle.

The Boat and Plane classes also inherit brand, model, and move() from Vehicle, but they both override the move() method.

Because of polymorphism we can execute the same method for all classes.

Python Dates

A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Import the datetime module and display the current date:

import datetime  
  
x = datetime.datetime.now()  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime1)

Date Output

When we execute the code from the example above the result will be:

2023-09-15 13:15:01.792721

The date contains year, month, day, hour, minute, second, and microsecond.

The datetime module has many methods to return information about the date object.

Here are a few examples, you will learn more about them later in this chapter:

Example

Return the year and name of weekday:

import datetime  
  
x = datetime.datetime.now()  
  
print(x.year)  
print(x.strftime("%A"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime2)

Creating Date Objects

To create a date, we can use the datetime() class (constructor) of the datetime module.

The datetime() class requires three parameters to create a date: year, month, day.

Example

Create a date object:

import datetime  
  
x = datetime.datetime(2020, 5, 17)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime3)

The datetime() class also takes parameters for time and timezone (hour, minute, second, microsecond, tzone), but they are optional, and has a default value of 0, (None for timezone).

ADVERTISEMENT

The strftime() Method

The datetime object has a method for formatting date objects into readable strings.

The method is called strftime(), and takes one parameter, format, to specify the format of the returned string:

Example

Display the name of the month:

import datetime  
  
x = datetime.datetime(2018, 6, 1)  
  
print(x.strftime("%B"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime)

A reference of all the legal format codes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Directive** | **Description** | **Example** | **Try it** |
| %a | Weekday, short version | Wed | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a) |
| %A | Weekday, full version | Wednesday | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a2) |
| %w | Weekday as a number 0-6, 0 is Sunday | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w) |
| %d | Day of month 01-31 | 31 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_d) |
| %b | Month name, short version | Dec | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b) |
| %B | Month name, full version | December | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b2) |
| %m | Month as a number 01-12 | 12 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m) |
| %y | Year, short version, without century | 18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y) |
| %Y | Year, full version | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y2) |
| %H | Hour 00-23 | 17 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_h2) |
| %I | Hour 00-12 | 05 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_i2) |
| %p | AM/PM | PM | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_p) |
| %M | Minute 00-59 | 41 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m2) |
| %S | Second 00-59 | 08 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_s2) |
| %f | Microsecond 000000-999999 | 548513 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_f) |
| %z | UTC offset | +0100 |  |
| %Z | Timezone | CST |  |
| %j | Day number of year 001-366 | 365 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_j) |
| %U | Week number of year, Sunday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u2) |
| %W | Week number of year, Monday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w2) |
| %c | Local version of date and time | Mon Dec 31 17:41:00 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_c) |
| %C | Century | 20 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_century) |
| %x | Local version of date | 12/31/18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x) |
| %X | Local version of time | 17:41:00 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x2) |
| %% | A % character | % | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_percent) |
| %G | ISO 8601 year | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_g) |
| %u | ISO 8601 weekday (1-7) | 1 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u) |
| %V | ISO 8601 weeknumber (01-53) | 01 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_v) |

Python JSON

[❮ Previous](https://www.w3schools.com/python/python_math.asp)[Next ❯](https://www.w3schools.com/python/python_regex.asp)

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

JSON in Python

Python has a built-in package called json, which can be used to work with JSON data.

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Import the json module:

import json

Parse JSON - Convert from JSON to Python

If you have a JSON string, you can parse it by using the json.loads() method.

The result will be a [Python dictionary](https://www.w3schools.com/python/python_dictionaries.asp).

Example

Convert from JSON to Python:

import json  
  
# some JSON:  
x =  '{ "name":"John", "age":30, "city":"New York"}'  
  
# parse x:  
y = json.loads(x)  
  
# the result is a Python dictionary:  
print(y["age"])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json)

Convert from Python to JSON

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

Example

Convert from Python to JSON:

import json  
  
# a Python object (dict):  
x = {  
  "name": "John",  
  "age": 30,  
  "city": "New York"  
}  
  
# convert into JSON:  
y = json.dumps(x)  
  
# the result is a JSON string:  
print(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python)

ADVERTISEMENT

You can convert Python objects of the following types, into JSON strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None

Example

Convert Python objects into JSON strings, and print the values:

import json  
  
print(json.dumps({"name": "John", "age": 30}))  
print(json.dumps(["apple", "bananas"]))  
print(json.dumps(("apple", "bananas")))  
print(json.dumps("hello"))  
print(json.dumps(42))  
print(json.dumps(31.76))  
print(json.dumps(True))  
print(json.dumps(False))  
print(json.dumps(None))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_all)

When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

Example

Convert a Python object containing all the legal data types:

import json  
  
x = {  
  "name": "John",  
  "age": 30,  
  "married": True,  
  "divorced": False,  
  "children": ("Ann","Billy"),  
  "pets": None,  
  "cars": [  
    {"model": "BMW 230", "mpg": 27.5},  
    {"model": "Ford Edge", "mpg": 24.1}  
  ]  
}  
  
print(json.dumps(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_all_in_one)

Format the Result

The example above prints a JSON string, but it is not very easy to read, with no indentations and line breaks.

The json.dumps() method has parameters to make it easier to read the result:

Example

Use the indent parameter to define the numbers of indents:

json.dumps(x, indent=4)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_indent)

You can also define the separators, default value is (", ", ": "), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values:

Example

Use the separators parameter to change the default separator:

json.dumps(x, indent=4, separators=(". ", " = "))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_separators)

Order the Result

The json.dumps() method has parameters to order the keys in the result:

Example

Use the sort\_keys parameter to specify if the result should be sorted or not:

json.dumps(x, indent=4, sort\_keys=True)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_json_from_python_sort_keys" \t "_blank)