Python Multithreading — Python's `threading` module allows to create threads as objects. In this tutorial, we shall learn how to work with threads in detailed sections.

- **Simple Example** — Multiple threads
- Create a thread
- Start a thread
- Pass arguments to thread function
- Is Thread Alive
- Thread Name

### Simple Example

We shall look into a simple example to threading module, and then go in detail of working with threads.

Note: The following examples are worked on environment with Python3 installed.

Following is a simple example to create multiple threads using `threading` module.

```python
import threading

def f():
    print('Thread function\n')
    return

for i in range(3):
    t = threading.Thread(target=f)
    t.start()
```

Output

```
Thread function
Thread function
Thread function
```

### Create a thread

You can create a thread in one of the two following ways.

1. Passing a method to `Thread` constructor.

```python
def f():
    print('Thread function\n')
    return

    t = threading.Thread(target=f)
```
Overriding run() method in a subclass of threading.Thread.

```python
import threading
class CustomThread(threading.Thread):
    def run(self):
        print('Custom thread function.
')

for i in range(3):
    t = CustomThread()
```

**Start a Thread**

A thread is started by applying start() method on the thread object.

```python
import threading
import time
def f():
    print('Thread running.
')
    return

# start threads by passing function to Thread constructor
for i in range(3):
    t = threading.Thread(target=f)
    t.start()
```

**Output**

Thread running.
Thread running.
Thread running.

**Passing arguments to the function supplied to Thread**

To pass arguments to the function supplied to Thread constructor, pass args in the Thread constructor as shown below:

```python
import threading
import time
def f(i):
    for p in range(3):
        time.sleep(i+1)
        print('Thread #',i,'\n')
        time.sleep()
    return

# start threads by passing function to Thread constructor
for i in range(3):
    t = threading.Thread(target=f, args=(i,))
    t.start()
```

**Output**


Is Thread Alive

threading.Thread.is_alive() could be used to check if the thread is alive or not.

Thread.is_alive() returns True if the thread is alive, False if not alive.

```python
import threading
import time

def f(i):
    time.sleep(i)
    return

# threads
t1 = threading.Thread(target=f, args=(1.2,), name="Thread#1")
t1.start()

t2 = threading.Thread(target=f, args=(2.2,), name="Thread#2")
t2.start()

for p in range(5):
    time.sleep(p*0.5)
    print('[',time.ctime(),']', t1.getName(), t1.is_alive())
    print('[',time.ctime(),']', t2.getName(), t2.is_alive())
```

Output

```
[ Tue Feb 27 17:58:54 2018 ] Thread#1 True
[ Tue Feb 27 17:58:54 2018 ] Thread#2 True
[ Tue Feb 27 17:58:55 2018 ] Thread#1 True
[ Tue Feb 27 17:58:55 2018 ] Thread#2 True
[ Tue Feb 27 17:58:56 2018 ] Thread#1 False
[ Tue Feb 27 17:58:56 2018 ] Thread#2 True
[ Tue Feb 27 17:58:57 2018 ] Thread#1 False
[ Tue Feb 27 17:58:57 2018 ] Thread#2 False
[ Tue Feb 27 17:58:59 2018 ] Thread#1 False
[ Tue Feb 27 17:58:59 2018 ] Thread#2 False
```

Thread Name

Thread name could be set or read using setName() and getName() methods.
In a function being called inside a thread, to get the current thread, use `threading.current_thread()`. In the following example we shall use this method to get current thread object.

```python
import threading
import time

def f(i):
    for p in range(3):
        time.sleep(i+1.5)
        print(threading.current_thread().getName())
    return

# start threads by passing function to Thread constructor
for i in range(3):
    t = threading.Thread(target=f, args=(i,))
    t.setName('Thread#' + str(i))
    t.start()
```

Output
---
Thread#0
Thread#1
Thread#0
Thread#2
Thread#0
Thread#1
Thread#2
Thread#0
Thread#2

### When does a Thread stops?

A thread stops when:

- `run()` method terminates normally. [or]
- an unhandled exception causes `run()` method to terminate abruptly.
<table>
<thead>
<tr>
<th>Python Programming</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Tutorial</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Get Started</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Python</td>
<td></td>
</tr>
<tr>
<td>Install Anaconda Python</td>
<td></td>
</tr>
<tr>
<td>Python HelloWorld Program</td>
<td></td>
</tr>
<tr>
<td>Python Variables</td>
<td></td>
</tr>
<tr>
<td>Python Variable Data Type Conversion</td>
<td></td>
</tr>
<tr>
<td>Python Comments</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Statements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Python If</td>
<td></td>
</tr>
<tr>
<td>Python If Else</td>
<td></td>
</tr>
<tr>
<td>Python While Loop</td>
<td></td>
</tr>
<tr>
<td>Python For Loop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Functions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advanced</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Multithreading</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Useful Resources</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Interview Questions</td>
<td></td>
</tr>
</tbody>
</table>