3. Quick Start

Start a preview for 10 seconds with the default settings:

```
import time
import picamera

camera = picamera.PiCamera()
try:
    camera.start_preview()
    time.sleep(10)
    camera.stop_preview()
finally:
    camera.close()
```

Note that you should always ensure you call `close()` on the PiCamera object to clean up resources. The following example demonstrates that Python's `with` statement can be used to achieve this implicitly; when the `with` block ends, `close()` will be called implicitly:

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.start_preview()
    time.sleep(10)
    camera.stop_preview()
```

The following example shows that certain properties can be adjusted "live" while a preview is running. In this case, the brightness is increased steadily during display:

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.start_preview()
    try:
        for i in range(100):
            camera.brightness = i
            time.sleep(0.2)
    finally:
        camera.stop_preview()
```

The next example demonstrates setting the camera resolution (this can only be done when the camera is not recording) to 640x480, then starting a preview and a recording to a disk file:

```python
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    camera.start_recording('foo.h264')
    camera.wait_recording(60)
    camera.stop_recording()
    camera.stop_preview()
```

The camera's default resolution is the display's resolution. If the display has been disabled (e.g. with *tvservice -o*), then the default resolution is 1280x720.

Note

Note that `wait_recording()` is used above instead of `time.sleep()`. This method checks for errors (e.g. out of disk space) while the recording is running and raises an exception if one occurs. If `time.sleep()` was used instead the exception would be raised by `stop_recording()` but only after the full waiting time had run.

This example demonstrates starting a preview, setting some parameters and then capturing an image while the preview is running:

```python
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (1280, 720)
    camera.start_preview()
    camera.exposure_compensation = 2
    camera.exposure_mode = 'spotlight'
    camera.meter_mode = 'matrix'
    camera.image_effect = 'gpen'
    # Give the camera some time to adjust to conditions
    time.sleep(2)
    camera.capture('foo.jpg')
    camera.stop_preview()
```

The following example customizes the Exif tags to embed in the image before calling `capture()`:

```python
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (2592, 1944)
    camera.start_preview()
    time.sleep(2)
    camera.exif_tags['IFD0.Artist'] = 'Me!'
    camera.exif_tags['IFD0.Copyright'] = 'Copyright (c) 2013 Me!'
    camera.capture('foo.jpg')
    camera.stop_preview()
```

See the documentation for `exif_tags` for a complete list of the supported tags.

The next example demonstrates capturing a series of images as a numbered series with a one minute delay between each capture using the `capture_continuous()` method:

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (1280, 720)
    camera.start_preview()
    time.sleep(1)
    for i, filename in enumerate(camera.capture_continuous('image{counter:02d}.jpg')):
        print('Captured image %s' % filename)
        if i == 100:
            break
        time.sleep(60)
    camera.stop_preview()
```

This example demonstrates capturing low resolution JPEGs extremely rapidly using the video-port capability of the `capture_sequence()` method. The framerate of the captures is displayed afterward:

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    start = time.time()
    camera.capture_sequence((
        'image%03d.jpg' % i
        for i in range(120)
        ), use_video_port=True)
    print('Captured 120 images at %.2ffps' % (120 / (time.time() - start)))
    camera.stop_preview()
```

This example demonstrates capturing an unencoded image in RGB format and producing a numpy array from the image:

```
import time
import picamera
import picamera.array

with picamera.PiCamera() as camera:
    with picamera.array.PiRGBArray(camera) as stream:
        camera.resolution = (1024, 768)
        camera.start_preview()
        time.sleep(2)
        camera.capture(stream, 'rgb')
        print(stream.array.shape)
```