Introduction
to
concurrency and multithreading

Concurrency

- Tasks are running in overlapping time periods → more usable program
- Non-deterministic
- Example:
  - Web browser: getting data from HTTP server, rendering the contents, responding to user input
- Can be done using single processor machine
- Not to be confused with parallelism

Parallelism

- Tasks are running at the same time → faster program
- Usually deterministic
- Example:
  - Using multiple CPUs (or cores) to solve complex calculation
- Requires hardware support

Concurrency implementation

- Multithreading: Multiple threads of control
- Using multiple operating system processes (fork)

Multithreading

- In application:
  - makes application more responsive
  - very useful when working with I/O bound operations (networking, file, database, etc)
- Lightweight, compared with creating new operating system process
- Faster context switching between threads, compared with between processes

Preemptive threading

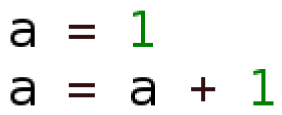
- Allows the scheduler to determine when a context switch should occur
- Offers better fault tolerance

Cooperative threading

- Also known as non-preemptive
- Scheduler never initiates a context switch: threads voluntary yield control
- Offers greater efficiency (on limited computing resources)

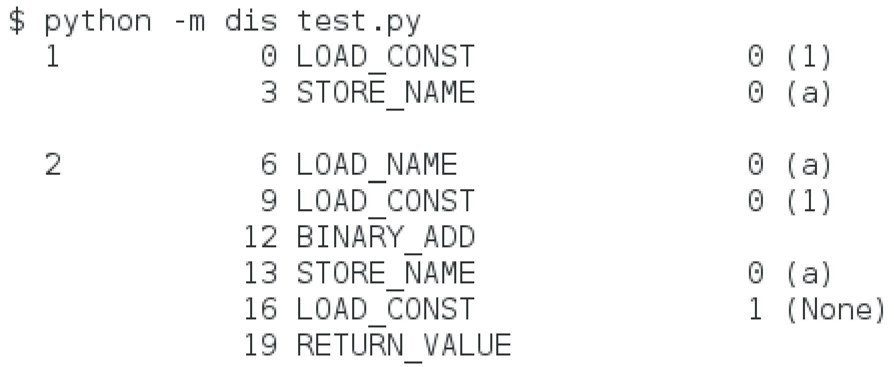
Note: atomic operation

Simple increment operation (in Python, saved as test.py):



```
a = 1
a = a + 1
```

Is actually not that simple:



```
$ python -m dis test.py
  1           0 LOAD_CONST               0 (1)
              3 STORE_NAME               0 (a)

  2           6 LOAD_NAME                0 (a)
              9 LOAD_CONST               0 (1)
             12 BINARY_ADD
             13 STORE_NAME               0 (a)
             16 LOAD_CONST               1 (None)
             19 RETURN_VALUE
```

It takes more than one bytecode (line 6 – 13). And, thread switch may happen.

Multithreading support in Java

- Thread support since version 1.0 (1996)
- Using green threads (emulated by VM) in early days. Switched to native threads support in Java 1.2 (1998)

Multithreading support in Python

- Modules: thread (or _thread in Python 3, low level), threading (high level), dummy_threading (when _thread module is not provided)
- Since version 3.2 (2011): concurrent.futures module, for simpler interface
- Green threads: greenlet, eventlet, gevent

Multithreading in GUI program

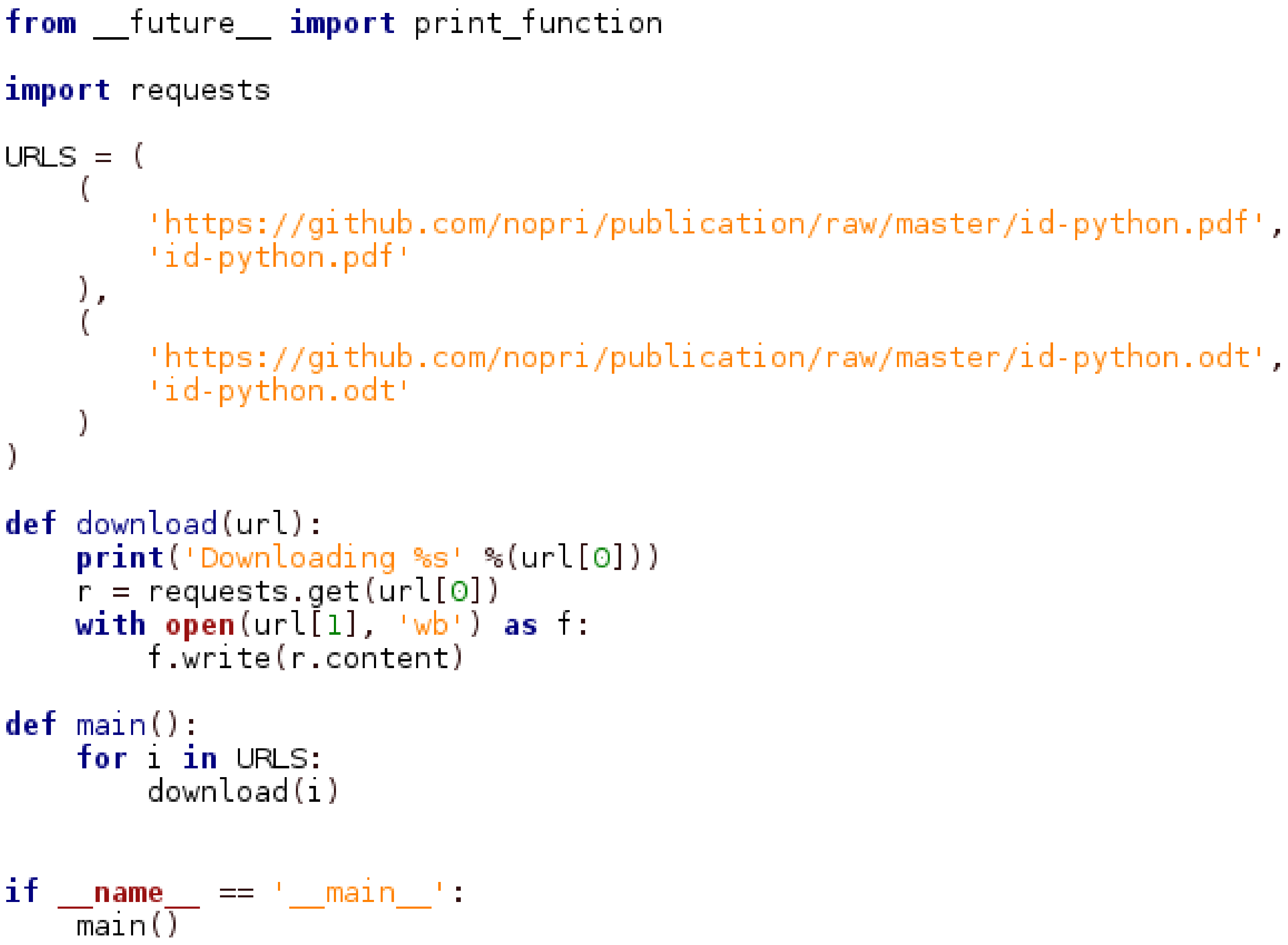
- Many popular GUI toolkits are not thread-safe: GTK+, Qt, Swing (Java)
- GTK+: only use GTK+ and GDK from the main thread
- Swing (Java): using Event Dispatch Thread (EDT)

Multithreading: Global Interpreter Lock in CPython

- Prevents multiple native threads from executing Python bytecodes at once
- Also prevents multithreaded program from taking full advantage of multiprocessor systems in certain situations
- This is not the limitation of Python (as a language)
- Note: there is one GIL per process

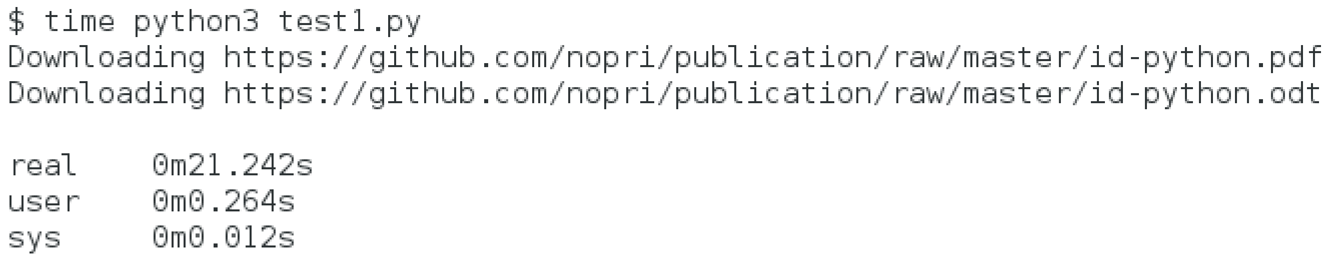
Source: https://wiki.python.org/moin/GlobalInterpreterLock

Single-threaded HTTP client



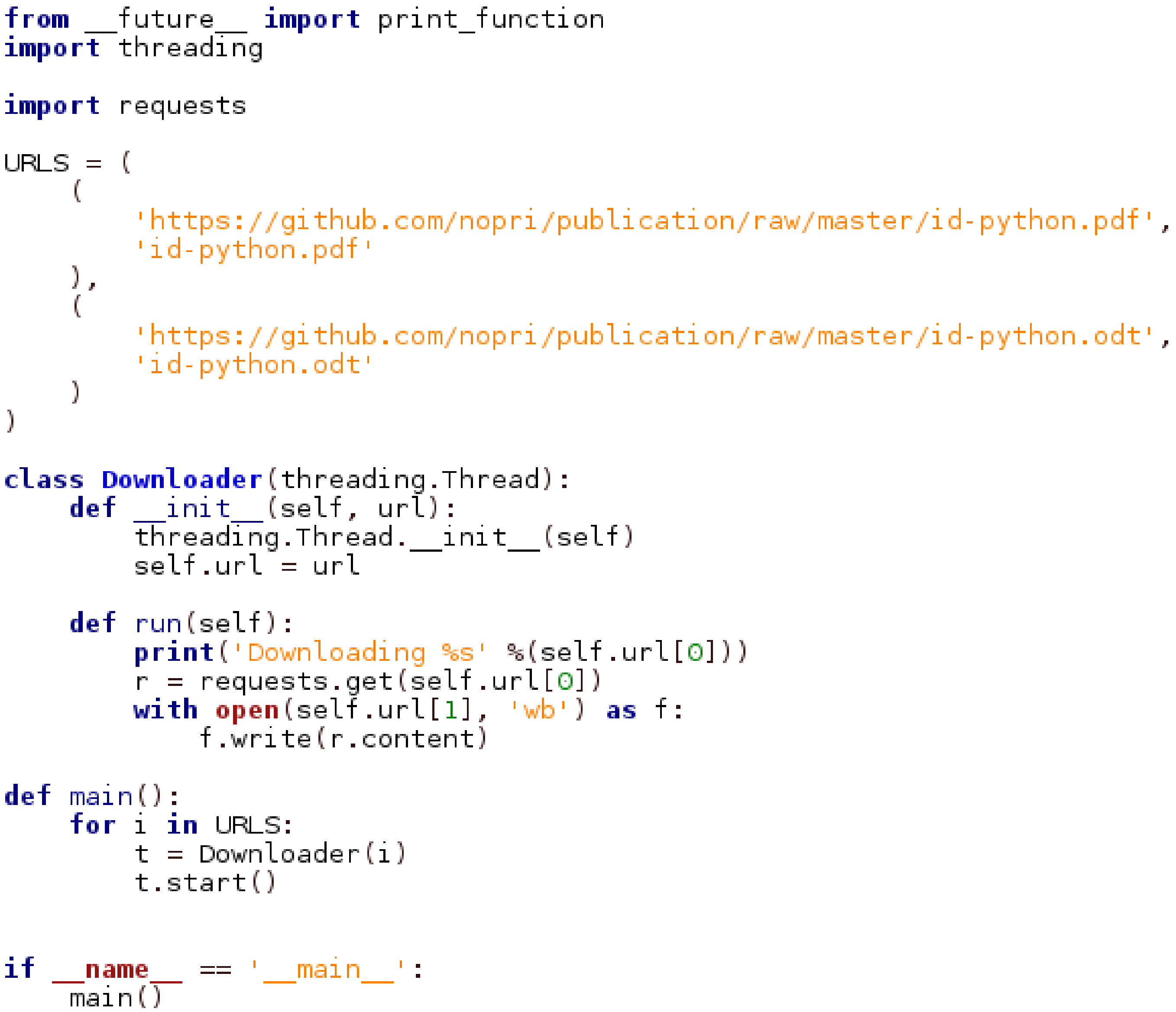
```python
from __future__ import print_function

import requests

URLS = (
    (
        'https://github.com/nopri/publication/raw/master/id-python.pdf',
        'id-python.pdf'
    ),
    (
        'https://github.com/nopri/publication/raw/master/id-python.odt',
        'id-python.odt'
    )
)

def download(url):
    print('Downloading %s' %(url[0]))
    r = requests.get(url[0])
    with open(url[1], 'wb') as f:
        f.write(r.content)

def main():
    for i in URLS:
        download(i)


if __name__ == '__main__':
    main()
```

Single-threaded HTTP client (2)



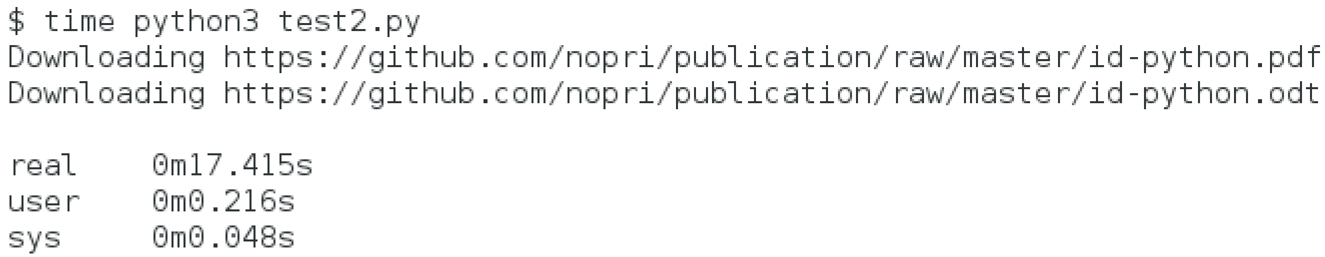
```
$ time python3 test1.py
Downloading https://github.com/nopri/publication/raw/master/id-python.pdf
Downloading https://github.com/nopri/publication/raw/master/id-python.odt

real    0m21.242s
user    0m0.264s
sys     0m0.012s
```

Multi-threaded HTTP client



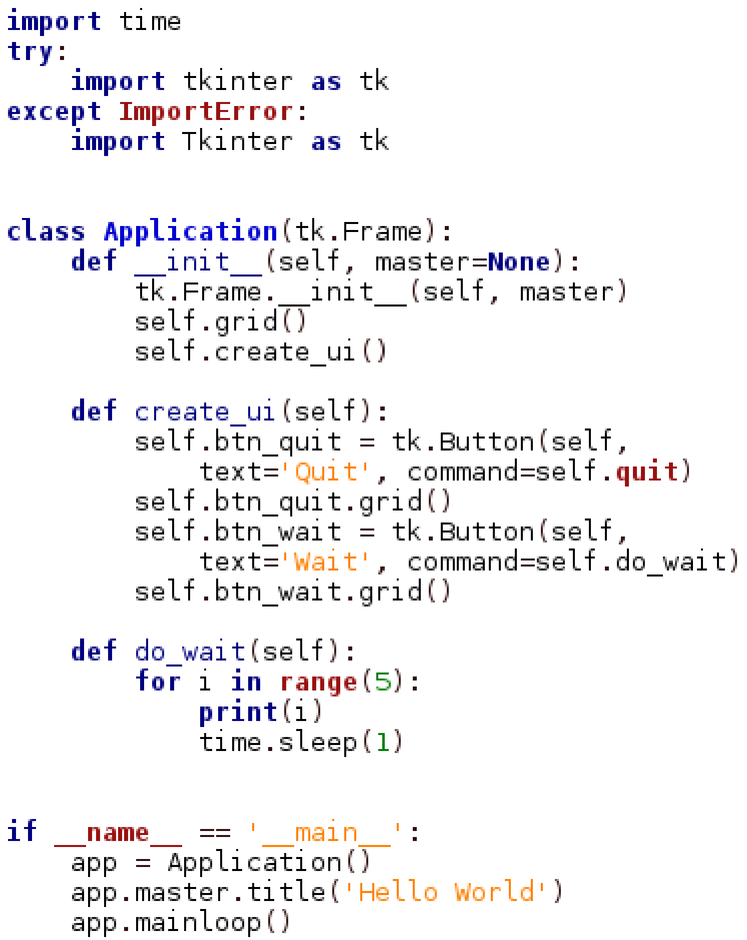
```python
from __future__ import print_function
import threading

import requests

URLS = (
    (
        'https://github.com/nopri/publication/raw/master/id-python.pdf',
        'id-python.pdf'
    ),
    (
        'https://github.com/nopri/publication/raw/master/id-python.odt',
        'id-python.odt'
    )
)

class Downloader(threading.Thread):
    def __init__(self, url):
        threading.Thread.__init__(self)
        self.url = url

    def run(self):
        print('Downloading %s' %(self.url[0]))
        r = requests.get(self.url[0])
        with open(self.url[1], 'wb') as f:
            f.write(r.content)

def main():
    for i in URLS:
        t = Downloader(i)
        t.start()


if __name__ == '__main__':
    main()
```

Multi-threaded HTTP client (2)

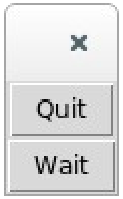


```
$ time python3 test2.py
Downloading https://github.com/nopri/publication/raw/master/id-python.pdf
Downloading https://github.com/nopri/publication/raw/master/id-python.odt

real    0m17.415s
user    0m0.216s
sys     0m0.048s
```

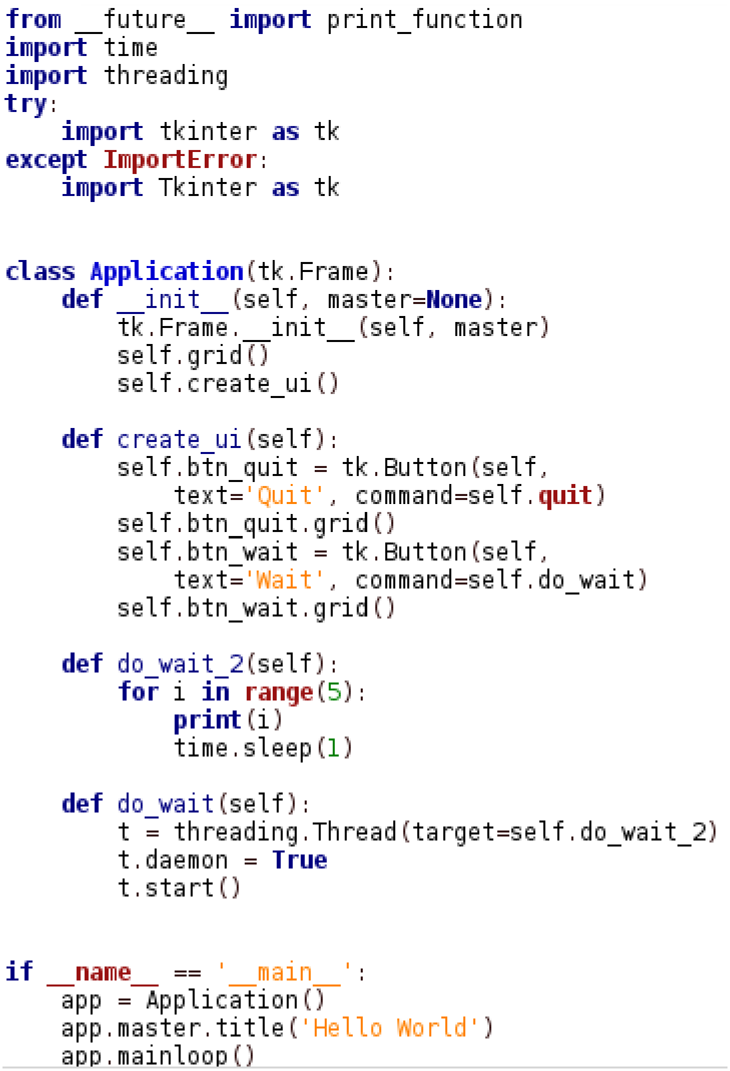
Unresponsive GUI



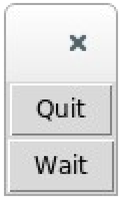
```
import time
try:
    import tkinter as tk
except ImportError:
    import Tkinter as tk


class Application(tk.Frame):
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.create_ui()

    def create_ui(self):
        self.btn_quit = tk.Button(self,
            text='Quit', command=self.quit)
        self.btn_quit.grid()
        self.btn_wait = tk.Button(self,
            text='Wait', command=self.do_wait)
        self.btn_wait.grid()

    def do_wait(self):
        for i in range(5):
            print(i)
            time.sleep(1)


if __name__ == '__main__':
    app = Application()
    app.master.title('Hello World')
    app.mainloop()
```



Responsive GUI



```
from __future__ import print_function
import time
import threading
try:
    import tkinter as tk
except ImportError:
    import Tkinter as tk


class Application(tk.Frame):
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.create_ui()

    def create_ui(self):
        self.btn_quit = tk.Button(self,
            text='Quit', command=self.quit)
        self.btn_quit.grid()
        self.btn_wait = tk.Button(self,
            text='Wait', command=self.do_wait)
        self.btn_wait.grid()

    def do_wait_2(self):
        for i in range(5):
            print(i)
            time.sleep(1)

    def do_wait(self):
        t = threading.Thread(target=self.do_wait_2)
        t.daemon = True
        t.start()


if __name__ == '__main__':
    app = Application()
    app.master.title('Hello World')
    app.mainloop()
```



Thank you