Lesson Plan 10 | Form 2 | Using Threads in Python

Objective

Students will be learn how to create a thread in Python and will receive visual proof that the OS schedules threads differently each time their program runs.

Warm-up

None.

Presentation

None.

Guided Practice

Walk students through the following program step-by-step on the whiteboard:

```
import threading

def squares():
  for i in range(1, 10):
    print “%d squared is: %d” % (i, i**2)
#squares()
t1 = threading.Thread(target=squares)
t1.start()

def cubes():
  for i in range(1, 10):
    print “%d cubed is %d” % (i, i**3)
#cubes()
t2 = threading.Thread(target=cubes)
t2.start()
```

Start by defining the *squares* function first and running it as a normal function. Then show them how to run it as a thread instead of the normal function call. Wait for students to suggest that we need to import *threading* if we want to use it.

Then define the *cubes* function, and as before, call it as a normal function. Ask the students: does the program display the same thing each time it runs? The answer is no due to OS scheduling; use this as an opportunity to ask students for this reason.

Independent Practice

If time allows, ask students to convert the normal function call for *cubes* into a thread.

Closing

Remind students of why we use threads (to do two or more things at the same time) even though we can never know exactly when a thread will run.

In Hindsight

02/08/2017: In both streams there were some students who immediately took to this, and there was even one student who successfully did the Independent Practice challenge without me asking her or suggesting it to the class, which was extremely impressive. I think the value of this lesson is in having the students see the output change each time – clearly there are things that are beyond their control or knowledge when using threads, and it’s important to impart that fact as clearly as possible.