Python Programming

Time 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

What is wrong with time()?

- time() provides real-world time (kind of) relative to a starting point
  - Good we can understand intuitively
  - It is maintained by the dedicated **hardware** on most computers
- The major issue with it is adjustable!
  - The clock can be **changed** by the system administrator
  - This makes it **unreliable**. Suddenly the time can **decrease**!
- Python has several other paths
  - clock (deprecated), perf_counter, monotonic
- The recommended one is perf_counter
- Future readings: link link

perf_counter

- It provides a **relative** time and has no reference time point
- It can be used only to measure time intervals
  - For more accurate results, we use **timeit module** (future)
    - Run a code like 1000000 times and average to know how much does it it takes
- It is not adjustable and administrator can’t affect it!

perf_counter

```
1   import time     # for .sleep
2   from time import perf_counter
3
4   start_time = perf_counter()
5
6   for i in range(5):
7       print(i)
8       time.sleep(1)   # hang for 1 second
9
10  end_time= perf_counter()
11  time_dif = end_time - start_time
12  print(time_dif)     # 5.003786797984503
13
14  # perf_counter_ns(): Py3.7: return time as nanoseconds
```

Future

- Some more extra functionalities in time module
- Proper handling for timezone (**pytz** module)
  - Datetime and Time modules are poor for timezone
  - Time Zones and Daylight savings time (DST)
- Datetime or time class?
  - If you are dealing with time zones issues, go datetime+pytz
- Reading Reading

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."