# Timing our code

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 1

# Timing our code

- How long does our code take to execute?
  - Depends from one language to another

# Timing our code

- How long does our code take to execute?
  - Depends from one language to another
- Python has a library `time` with various useful functions

# Timing our code

- How long does our code take to execute?
  - Depends from one language to another
- Python has a library `time` with various useful functions
- `perf_time()` is a performance counter
  - Absolute value of `perf_time()` is not meaningful
  - Compare two consecutive readings to get an interval
  - Default unit is seconds

# Timing our code

- How long does our code take to execute?
  - Depends from one language to another
- Python has a library `time` with various useful functions
- `perf_time()` is a performance counter
  - Absolute value of `perf_time()` is not meaningful
  - Compare two consecutive readings to get an interval
  - Default unit is seconds

```
import time

start = time.perf_counter()
...
# Execute some code
...
end = time.perf_counter()

elapsed = end - start
```

# A timer object

- Create a timer class

```
import time
class Timer:
```

# A timer object

- Create a timer class
- Two internal values
  - _start_time
  - _elapsed_time

```
import time
class Timer:

  def __init__(self):
    self._start_time = 0
    self._elapsed_time = 0
```

# A timer object

- Create a timer class
- Two internal values
  - `_start_time`
  - `_elapsed_time`
- `start` starts the timer

```
import time
class Timer:

  def __init__(self):
    self._start_time = 0
    self._elapsed_time = 0

  def start(self):
    self._start_time = time.perf_counter()
```

# A timer object

- Create a timer class
- Two internal values
  - `_start_time`
  - `_elapsed_time`
- `start` starts the timer
- `stop` records the elapsed time

```
import time
class Timer:

  def __init__(self):
    self._start_time = 0
    self._elapsed_time = 0

  def start(self):
    self._start_time = time.perf_counter()

  def stop(self):
    self._elapsed_time =
       time.perf_counter() - self._start_time

  def elapsed(self):
    return(self._elapsed_time)
```

# A timer object

- Create a timer class
- Two internal values
  - `_start_time`
  - `_elapsed_time`
- `start` starts the timer
- `stop` records the elapsed time
- More sophisticated version in the actual code

```
import time
class Timer:

  def __init__(self):
    self._start_time = 0
    self._elapsed_time = 0

  def start(self):
    self._start_time = time.perf_counter()

  def stop(self):
    self._elapsed_time =
       time.perf_counter() - self._start_time

  def elapsed(self):
    return(self._elapsed_time)
```

# A timer object

- Create a timer class
- Two internal values
  - `_start_time`
  - `_elapsed_time`
- `start` starts the timer
- `stop` records the elapsed time
- More sophisticated version in the actual code
- Python executes $10^7$ operations per second

```
import time
class Timer:

  def __init__(self):
    self._start_time = 0
    self._elapsed_time = 0

  def start(self):
    self._start_time = time.perf_counter()

  def stop(self):
    self._elapsed_time =
       time.perf_counter() - self._start_time

  def elapsed(self):
    return(self._elapsed_time)
```