# Python Programming

# Circular Imports 1

**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)

# Circular Imports

- We learned python imports a module only once!
- The interpreter checks the module registry in sys.modules
  - Is it cached?
    - Yes ⇒ Use it
    - No ⇒ **1) Mark** in the registry. **2) initialize it**
      - Observe it will be marked although it is partially initialized so far
- But when if module A is importing module B which is importing module A?
  - A ⇒ B ⇒ A Cycle: Circular Import
  - It can be a bigger tricky cycle: **A** ⇒ B ⇒ C ⇒ D ⇒ **A**
- This cycle typically causes problems
  - Python itself won't reimport (stops normally - doesn't go forever)
  - But some functions/classes might not be **defined**

# Let’s run c.py

a.py

```
1  import b
2  def af():
3      return b.x
4  af()
```

b.py

```
1  import a
2  x = 1
3  def bf():
4      print(a.af())
```

c.py

```
1  import a
2
```

- c.py
- c.py: line 1
  - …
  - a.py: line 1
    - …
    - b.py line 1
    - b.py line 2
    - b.py line 3
  - a.py: line 2
  - a.py: line 4
  - a.py: line 3
- c.py: line 2

- Run as a **script**
- import a
  - Is initialized? No. Mark it & initize
  - import b
    - Is initialized? No. Mark it & initize
    - Import a: in-progress - **skip**
    - x = 1
    - def bf():
  - def af()
  - call af()
  - b.x: [get from the module] ⇒ 1

# Let’s run d.py

a.py

```
1  import b
2  def af():
3      return b.x
4  af()
```

b.py

```
1  import a
2  x = 1
3  def bf():
4      print(a.af())
```

d.py

```
1  import b
2
```

- d.py
- d.py: line 1
  - …
  - b.py: line 1
    - …
    - a.py line 1
    - a.py line 2
    - a.py line 4
    - a.py line 3

- Run as a **script**
- import b
  - Is initialized? No. Mark it & initize
  - import a
    - Is initialized? No. Mark it & in
    - Import b: in-progress - **skip**
    - def af()
    - call af()
    - b.x: Error! **partially initialized module 'b' has no attribute 'x'**
- Observe:
  - Running b.py itself won’t cause error
  - As b.py in first run is not marked
    - Script here / NOT imported

# Another case

x.py

```
1  import y
2
3  def f1():
4      y.f2()
5
6  def f3():
7      pass
8
```

y.py

```
1  import x
2
3  def f2():
4      # Move the above import here
5      x.f3()
```

z.py

```
1  import x
2
3  x.f1()
4
5  import y
6  y.f2()
7
```

- It works fine.
- When we call functions, all modules will be fully initialized

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."