# 太极图形课

01讲 答疑

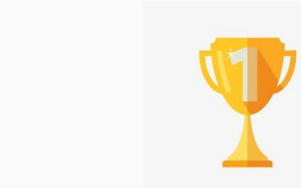
taichi

# 并行与串行

- 搬砖

# 并行与串行

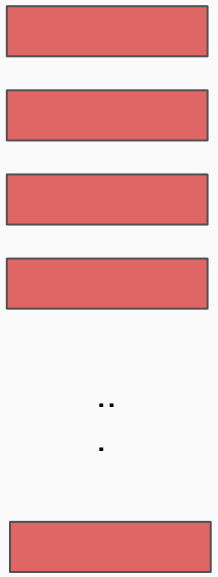
- 搬砖
  - 目标：1000,000块砖需要搬
  - 方法：
    - 串行：只有 1个大力士 (CPU)，每次可以搬100块砖
    - 并行：有100只猴子 (GPU)，每只猴子可以搬10块砖
  - 问题：谁搬的更快？
    - 答案：
      - 大力士：100，0000/100　= 10,000 次
      - 猴子们：100,0000/100/10 =　1,000 次

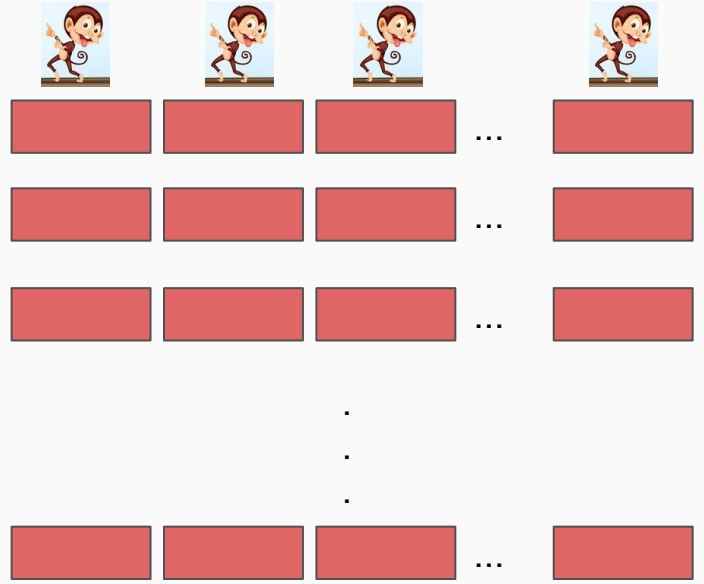


# ti.kernel, ti.func

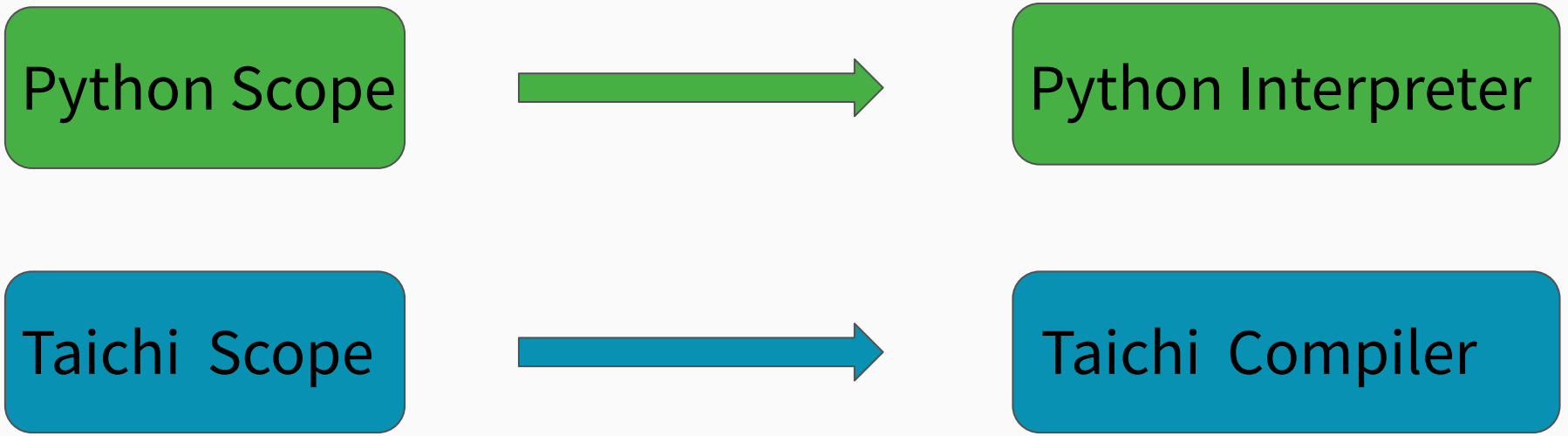
```python
def banzhuan():
    for i in range(len(zhuan)):
      move(zhuan[i])
```

```python
@ti.kernel
def banzhuan():
    for i in zhuan:
        move(zhuan[i])
```





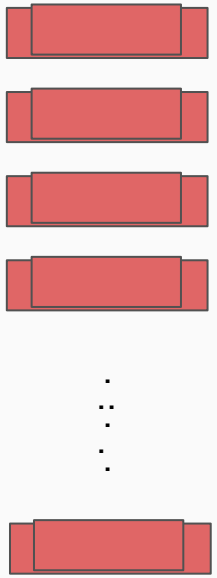
# Python Scope  vs Taichi Scope

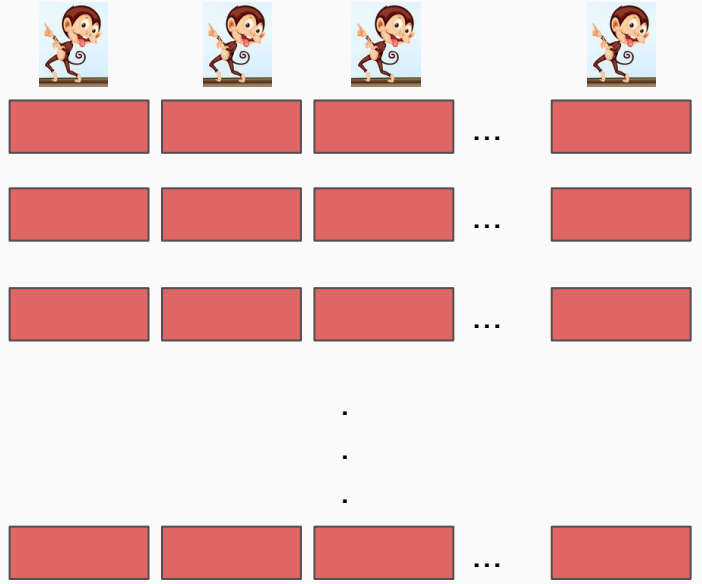


# 回顾 ti.kernel, ti.func

```python
def banzhuan():
    for i in range(len(zhuan)):
      move(zhuan[i])
```

```python
@ti.kernel
def banzhuan():
    for i in zhuan:
        move(zhuan[i])
```

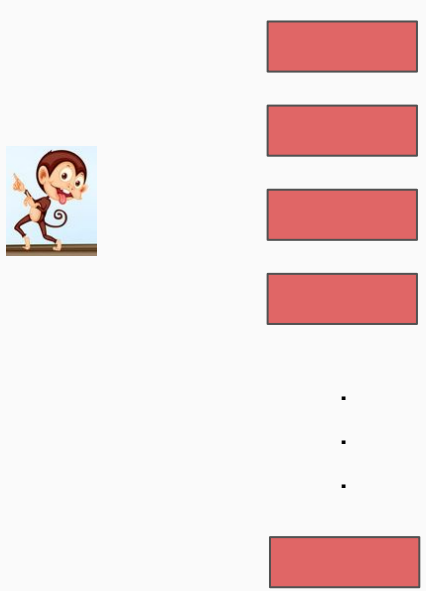


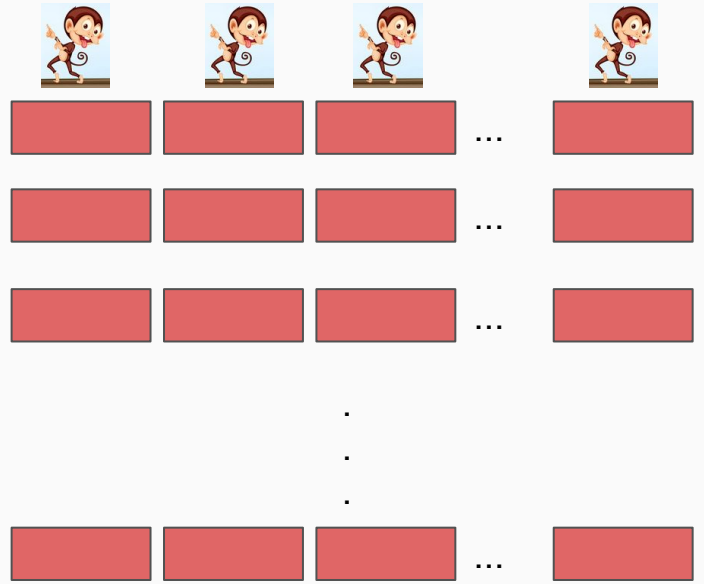


# 回顾 ti.kernel, ti.func

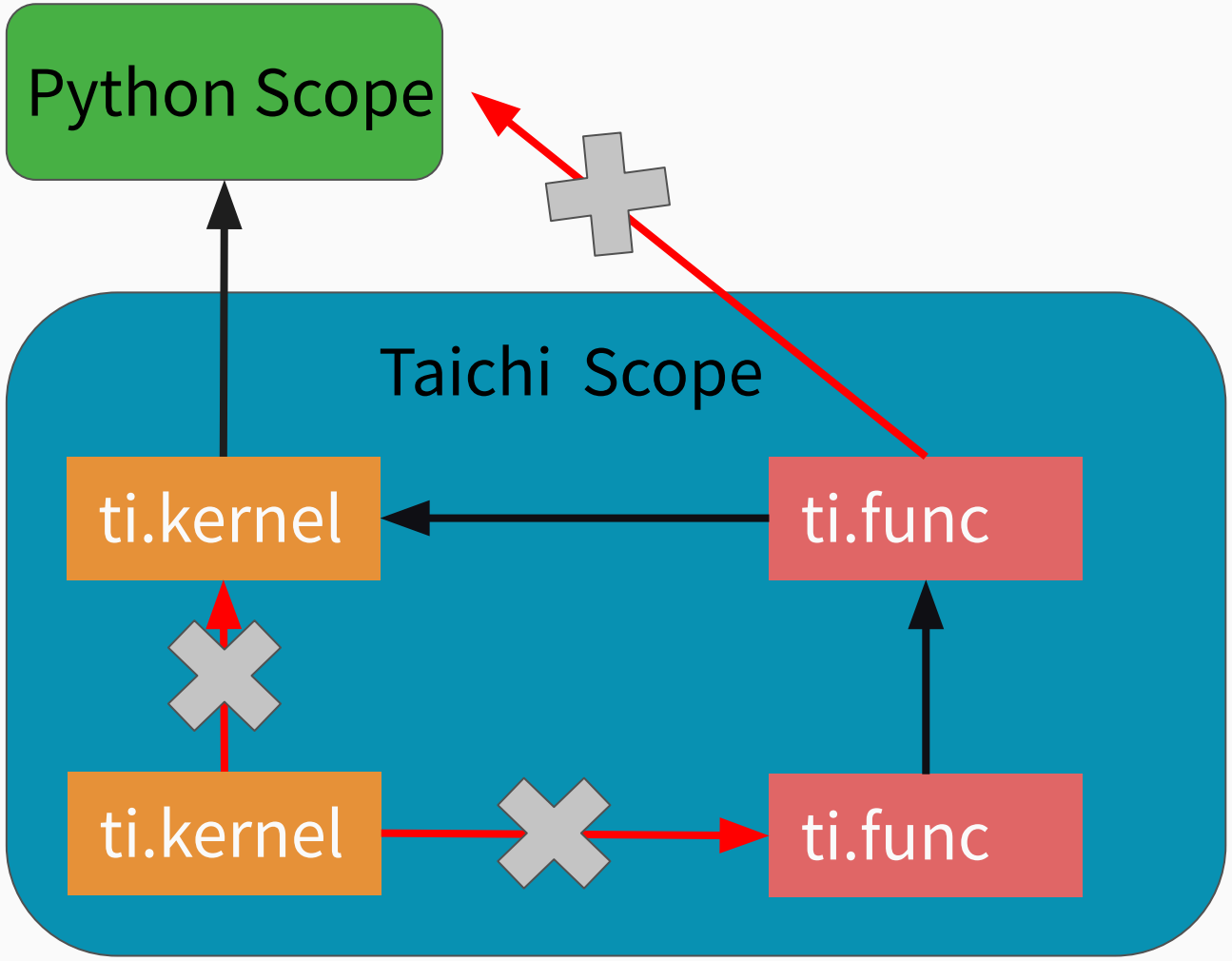
```python
def banzhuan():
    for i in range(len(zhuan)):
      move(zhuan[i])
```

```python
@ti.kernel
def banzhuan():
    for i in zhuan:
        move(zhuan[i])
```





# 函数调用


Python Scope
Taichi Scope
ti.kernel
ti.func
ti.kernel
ti.func

# ti.field, ti.kernel, ti.func


**Data**

ti.field
ti.Vector.field
ti.Matrix.field
ti.Struct.field

...

**Computation**

ti.kernel
ti.func

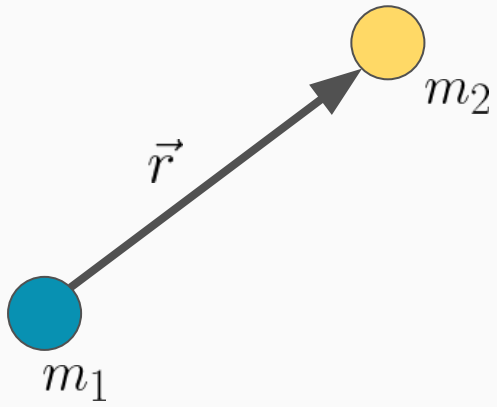
taichi

# N-body

- 初始化
  - 随机数生成: ti.random $\in [0, 1]$
  - 位置和速度的设置 : 圆的参数表示

$$y = r \cdot (sin(\theta), cos(\theta))$$

- 计算
  - 万有引力公式




$$f = \frac{Gm_1m_2}{r^2}\frac{\vec{r}}{\|r\|} = \frac{Gm_1m_2}{r^3}\vec{r}$$

# N-body

- 计算
  - For 循环: load balance and memory footprint

```python
for i in range(5):
    p = pos[i]
    for j in range(i):
        diff = p-pos[j]                      # 1: read
        r = diff.norm(1e-5)                  #0.5: compute
        f = -G * m * m * (1.0/r)**3 * diff #0.5: compute
        force[i] += f    # 5: atomic add: read and write
        force[j] += -f   # 5: atomic add: read and write
```

| Thread 1 | Thread 2 | Thread 3 | Thread 4 | Thread 5 |
|---|---|---|---|---|
| | (1, 0) | (2, 0) | (3, 0) | (4, 0) |
| | | (2, 1) | (3, 1) | (4, 1) |
| | | | (3, 2) | (4, 2) |
| | | | | (4, 3) |
| 0 | 12 | 24 | 36 | 48 |

work: 120

```python
for i in range(5):
    p = pos[i]
    for j in range(5):
        if i != j:
            diff = p-pos[j]                      #  1: read
            r = diff.norm(1e-5)                  #0.5: compute
            f = -G * m * m * (1.0/r)**3 * diff  #0.5: compute
            force[i] += f # 5: atomic add: read and write
```

| Thread 1 | Thread 2 | Thread 3 | Thread 4 | Thread 5 |
|---|---|---|---|---|
| (0, 1) | (1, 0) | (2, 0) | (3, 0) | (4, 0) |
| (0, 2) | (1, 2) | (2, 1) | (3, 1) | (4, 1) |
| (0, 3) | (1, 3) | (2, 3) | (3, 2) | (4, 2) |
| (0, 4) | (1, 4) | (2, 4) | (3, 4) | (4, 3) |
| 28 | 28 | 28 | 28 | 28 |

work: 140

taichi