```python
if __name__ == "__main__":

    clear_terminal_ansi()

    start_time = time.time()

    m = multiprocessing.Manager()
    res = m.list()
    nums = [1_822_725, 22_059_421, 32_374_695,
    88_754_320, 97_162_66, 200_745_654]
    num = nums[-1]
    cpu_count = multiprocessing.cpu_count()
    rg = num//cpu_count
    inputs= [(i*rg,rg ,res) for i in range(cpu_count) ]
    inputs.append( (cpu_count*rg,num%cpu_count,res) )
    print(num%cpu_count)

    ps = [multiprocessing.Process(target=approximate_pi,args=i) for i in inputs]

    for p in ps:
        p.start()

    for p in ps:
        p.join()
        p.close()

    pi =2.0
    print(res)
    for r in res:
        pi*=r

    print(pi)

    end = time.time()
    print(end - start_time)
```

```python
import multiprocessing
import time
def clear_terminal_ansi():
    print('\033c',end='')


def get_nom_den(x):
        nom = 2.0*(x//2 +1 )
        den = 2.0*((x-1)//2 +1 ) +1
        return nom, den

def approximate_pi(init,rg,res):

    pi_2=1.0
    nom,den = get_nom_den(init)
    for i in range(init,init + rg):
        pi_2 *= nom / den
        if i % 2:
                nom += 2
        else:
                den += 2

    res.append(pi_2)
```

I used multiprocessing because there is no need to share variables and this is good for multiprocessing also python has only one main thread so in this case, it is better to use multiprocessing.
By doing so I make the code execution 3 times faster. 27s -> 9s

```
3.141592645761164
27.654179573059082
```

```
3.1415926430226575
9.46540904045105
```

```python
from numba import jit
@jit(nopython=True)
def approximate_pi(n):
    pi_2 = 1.0
    nom, den = 2.0, 1.0
    for i in range(n):
        pi_2 *= nom / den

        if i % 2:
            nom += 2
        else:
            den += 2

    return 2*pi_2
```

By using numba and jit decorator i made code excution 32 times faster.

```
3.141592645761164
0.8629465103149414
```

```python
import numpy as np
from skimage import data, color
from skimage.transform import resize
imgs = np.uint8(data.lfw_subset()*255)

def res_skimage(imgs):
    new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
    imgs = np.moveaxis(imgs,0,-1)
    image_resized = resize(imgs, new_size, anti_aliasing=True,)

    return np.asarray(image_resized)

%lprun -f res_skimage res_skimage(imgs)
```

0.0s

```
%lprun -f res_skimage res_skimage(imgs)
```

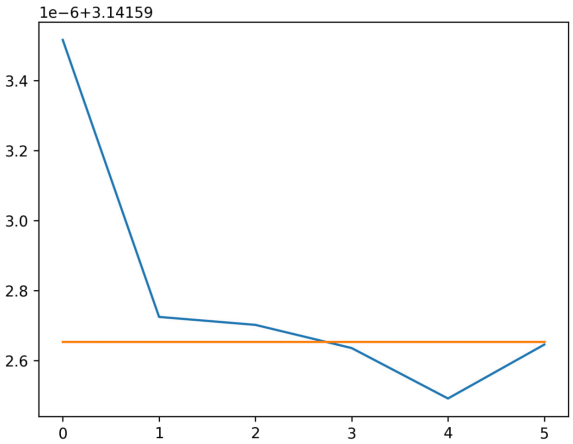
0.5s

```
Timer unit: 1e-07 s

Total time: 0.476469 s
File: C:\Users\Majid\AppData\Local\Temp\ipykernel_19192\1285670587.py
Function: res_skimage at line 6

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     6                                           def res_skimage(imgs):
     7         1        354.0    354.0      0.0      new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
     8         1         15.0     15.0      0.0      res_im = []
     9       201      59147.0    294.3      1.2      for im in imgs:
    10       200    4581638.0  22908.2     96.2          image_resized = resize(im, new_size, anti_aliasing=True)
    11       200      42590.0    212.9      0.9          res_im.append(image_resized)
    12
    13         1      80948.0  80948.0      1.7      return np.asarray(res_im)
```

resizing each image one by one in a for loop is really time-consuming so it is better to stack all of them on channel dim and resize once that is what I did with np. moveaxis. after that execution time becomes 0.01 instead of 0.47.




```
Timer unit: 1e-07 s

Total time: 0.0151691 s
File: C:\Users\Majid\AppData\Local\Temp\ipykernel_19192\3764382893.py
Function: res_skimage at line 6

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     6                                           def res_skimage(imgs):
     7         1        166.0    166.0      0.1      new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
     8         1       1194.0   1194.0      0.8      imgs = np.moveaxis(imgs,0,-1)
     9         1     150258.0 150258.0     99.1      image_resized = resize(imgs, new_size, anti_aliasing=True)
    10
    11         1         73.0     73.0      0.0      return np.asarray(image_resized)
```

Majid Sharghi Foroushani, M.N. 23193769, ix05ogym