```python
import numpy as np
import pandas as pd


def conf_interval(estimate, std_error, conf):
    '''
    estimate: the estimated vallue to build an interval around
    std_error: standard error
    conf: percent confidence as a float (e.g. .95 for 95%)
    '''
    z_dict = {.95: 1.96} # maybe i'll fill out this dict someday
    z = z_dict[conf]
    bottom = estimate-z*std_error
    top = estimate+z*std_error

    return (bottom, top)


'''Create a function insidecircle that takes two inputs between 0 and 1
and returns 1 if these points fall within the unit circle.'''
def insidecircle(x, y):
    if (x**2+y**2) <= 1:
        return 1
    return 0  # else


''' Create a function estimatepi that takes a single input N, generates N pairs of
uniform random numbers and uses insidecircle to produce an estimate of π as described
above.
In addition to the estimate of π, estimatepi should also return the standard error of
this estimate, and a 95% confidence interval for the estimate. '''
def estimatepi(N):
    result = {}
    xa = np.random.uniform(0,1,N)
    ya = np.random.uniform(0,1,N)
    pairsa = zip(xa, ya)

    inside_count = 0
    for pair in pairsa:
        inside_count += insidecircle(*pair)

    result['pi_est'] = 4*inside_count/N
    result['pi_err'] = 4*np.sqrt(inside_count/N * (1 - inside_count/N) / N)
    result['interval'] = conf_interval(result['pi_est'], result['pi_err'], .95)

    return result

def iter_N(start, end, interval):
    df = pd.DataFrame()
    N = start
    goal = None
    interv = None
    while N <= end:
        result = estimatepi(N)
        print('current N:', N)
        print('\testimate of pi: {} \n\terror: {}\n\t95% confidence interval:
        {}'.format(*result.values()))
        if np.pi - result['interval'][0] <= .1 and \
            result['interval'][1] - np.pi <= .1:
            if not goal:
                goal = N
                interv = result['interval']

        else:
            goal = None

        df = df.append(result, ignore_index=True)
```

```
64          N = N + interval
65
66      return df, goal, interv
67
68
69  if __name__ == "__main__":
70      print('b) testing N=1000')
71      # solution to b)
72      throwaway = iter_N(1000, 1000, 1)
73
74      print('\n\n----------\nc) iterating N')
75      # solution to c)
76      df_c, goal, interv = iter_N(1000, 10000 , 500)
77      print(df_c)
78      print('ensure within .1:', goal)
79
80      print('\n\n----------\nd) collecting 500 at goal N={}'.format(goal))
81      df_d = pd.DataFrame()
82      for i in range(500):
83          result = estimatepi(goal)
84          df_d = df_d.append(result, ignore_index=True)
85
86      print('std deviation', df_d.pi_est.std())
87      perc_within = len(df_d[(df_d.pi_est >= interv[0]) & (df_d.pi_est <= interv[1])]) /
        len(df_d)
88      print('Percent of esitmates falling within interval: {}'.format(perc_within))
89
90
91
92
```