```
"""IS597PR (Instructor Weible) - Assignment 1

CORRECT all syntax & formatting mistakes in this program so it works properly.

Brute-force (inefficient) solver of the "Monkey and the Coconuts" puzzle.

This is an example of a "Diophantine problem" -- one which implies a system of
algebraic equations where only integer solution(s) are valid.

I learned of it from the opening chapter of "The Colossal Book of Mathematics"
by Martin Gardner. Ben Williams' popularized the problem with these parameters
in a Saturday Evening Post story:

    Five men and a monkey were shipwrecked on an island. They spent the first
    day gathering coconuts. During the night, one man woke up and decided to
    take his share of the coconuts. He divided them into five equal piles. One
    coconut was left over so he gave it to the monkey, then hid his share,
    put the rest back together, and went back to sleep.

    Soon a second man woke up and did the same thing. After dividing the
    coconuts into five equal piles, one coconut was left over which he gave to
    the monkey. He then hid his share, put the rest back together, and went to
    bed. The third, fourth, and fifth man followed exactly the same procedure.

    The next morning, after they all woke up, they divided the remaining
    coconuts into five equal shares. This time no coconuts were left over.

    How many coconuts were there in the original pile?

See https://en.wikipedia.org/wiki/The_monkey_and_the_coconuts for a detailed
history, analysis, and multiple ways this problem can be solved.
"""
```

This is a comment.

```
Import sys

def solve_coconut_problem(number_of_men, monkey_gets_coconut_at_end,
                          maximum_number_to_try, verbose=False):
```

This starts defining a new function.

```
"""This algorithm starts at the beginning of the problem by trying numbers
of coconuts before nightfall, and going through the divisions and thefts
to see if it results in integers all the way to an equal portion in
the morning. If so, a solution was found and will be printed and returned.

:param number_of_men: how many men successively sneak "their share" during the night
:param monkey_gets_coconut_at_end: False or True, whether 1 coconut gets left for the monkey
:param maximum_number_to_try: the highest number of starting coconuts to test.
:param verbose: whether to print details while searching.
:return: a list of solution(s) found [the starting number(s) of coconuts]
"""
```

this big comment is a "docstring".

```
solutions_found = [];

# guess represents how many coconuts we might have started with
for guess in range(0, maximum_number_to_try):
if verbose:
    print('Testing a start of {} coconut(s).  '.format(guess))
```

```
coconuts = guess

ok_so_far = True
for man in range(1, number_of_men + 1):
if coconuts % number_of_men != 1:
# quantity was not evenly divisible with a single remainder.
if verbose print('Wrong because man #{} would have found {} coconuts\n'.format(man, coconuts))
ok_so_far = False
break

 equal_share = coconuts // number_of_men
 coconuts -= equal_share
 coconuts -= 1  # The monkey's coconut

 if verbose:
 print('man {} took {}, gave monkey 1, and {} remain.'.format(
    man, equal_share, coconuts
 ))

  if ok_so_far:
  # check final conditions match for "the next morning":
  if (coconuts % number_of_men == 1 and monkey_gets_coconut_at_end) or \
     (coconuts % number_of_men == 0 and not monkey_gets_coconut_at_end):
  print('Solution found. They could have started with {:10,} coconuts.'.format(guess))
  solutions_found.append(guess)
  continue

  if not solutions_found:
     print("No solutions found with fewer than {} starting coconuts.".format(maximum_number_to_try))
  return solutions_found
```

The function definition ends here.

```
   if __name__ == '__main__':
      print("Running Python:", sys.version_info, "\n")
      print("Searching for solutions using William's variation:")
      solve_coconut_problem(number_of_men=5, monkey_gets_coconut_at_end=False,
                     maximum_number_to_try=20000)


      try {
      print("\nNow try a custom variation:")
      sailors = int(input("How many sailors are on your island? "))
      leftover = input("Is there one coconut left for the monkey at the end? ")
      if leftover.upper().strip() in ['T', 'Y', 'TRUE', 'YES']:
         leftover = True
      else:
         leftover = False
      solve_coconut_problem(number_of_men=sailors,
                     monkey_gets_coconut_at_end=leftover,
                     maximum_number_to_try=10000)

      }
      except ValueError {
      print("Invalid input, try again.\n")
      }
```