Question 1. In each of these, rewrite the computation of result in a simpler or more elegant way.

a. \[ a = b == False \] # assume b is a boolean value

b. 
\[
result = True \text{ if } \text{randint}(0,1) == 0 \text{ else } False \] # assume randint imported

c. 
\[
result = False \\
\text{for } c \text{ in } s: \ \text{# s is a string} \\
\quad \text{if } c == '!': \\
\quad \quad result = True
\]

d. 
\[
result = '' \\
\text{for } c \text{ in } s: \ \text{# s is a string} \\
\quad result = c + result
\]

e. 
\[
result = [] \\
\text{for } r \text{ in } \text{range}(100): \\
\quad result.append( r*r*pi ) \quad \text{# assume pi is an imported math constant}
\]
Question 2. Answer the following with reasonably short, concise answers.

a. Computer scientists are big fans of abstraction. What is abstraction?

How do modules support abstraction?

b. Python is an object-oriented language. Indeed, everything in Python is an object. Variables refer to objects. When we assign one variable to another \((a = b)\) what actually happens?

c. Some objects in Python are immutable; others are not. A string (str), for example, is immutable. What is another immutable type? or, for more points: Why is the immutability of strings useful?

What is an example of a mutable type? or, for more points: What is sort stability?

d. Suppose we have a function, factors(n), which works hard to develop a list of integers that evenly divide \(n\). If we memoize factors(n) what are we doing?

e. Suppose we have a script, horoscope.py, that prints our horoscope to standard output. In the shell, how would we capture the output of horoscope.py to a file, thursday?

*