1. According to CU's website, each student gets an *Identikey* consisting of the first two letters of their first name, followed by the first to letters of their last names, and ending with four random digits.

   Write a function called `identikey` that takes two strings called `first` and `last`, corresponding to a student's first and last names, and returns a valid Identikey string for that student.

   Solution:

```
def identikey(first, last):
   digit1 = random.randint(0, 9)
   digit2 = random.randint(0, 9)
   digit3 = random.randint(0, 9)
   digit4 = random.randint(0, 9)
   return first[0:2] + last[0:2] + str(digit1) + str(digit2)
                                 + str(digit3) + str(digit4)
```

2. For the following four problems, write down what each code block would display if executed in a Jupyter cell.

   (a)
```
(242 % 2 == 0) and (31 // 3 == 9)
```

   (b)
```
alist = [1]
for n in range(3):
    alist.append(1 + 2 * alist[n])
alist
```

   (c)
```
[k ** 2 for k in range(-2, 2) if k!= 0]
```

   (d)
```
first = 'matt'
for ltr in first:
    print(2 * ltr + '!')
```

   Solution:

   (a)
```
False
```

   (b)
```
[1, 3, 7, 15]
```

   (c)
```
[4, 1, 1]
```

   (d)
```
mm!
aa!
tt!
tt!
```

3. An integer m is called a *divisor* of another integer n if m divides n evenly. In other words, m is a divisor of n if n/m is an integer. For example, the positive divisors of 6 are 1, 2, 3, and 6.

   The *proper divisors* of the positive integer n are those positive divisors of n which are **not** equal to n. For example, the proper divisors of 6 are 1, 2, and 3.

(a) Write a function called `prop_div` which takes a positive integer `n` as input, and returns a list of the proper divisors of `n`.

(b) A *perfect number* is any integer n which is equal to the sum of its proper divisors. Define a function `is_perfect` which takes a **any** integer `n` as input and returns `True` if `n` is a perfect number, `False` if `n` is not a perfect number, and the string `'error'` if `n` is not a positive integer.

You may use the function from (a) in (b), if you find that useful.

Solution:

(a)
```
def prop_div(n):
    return [k for k in range(1, n) if n % k == 0]
```

(b)
```
def is_perfect(n):
    if n <= 0:
        return 'error'
    else:
        return sum(prop_div(n)) == n
```

4. Write a function `digit_sum` which takes a positive three-digit integer as input and returns the sum of those digits. For example, `digit_sum(123)` will return 6 because $1 + 2 + 3 = 6$. Use **only** arithmetic operations, **not** string operations.

Solution:

```
def digit_sum(n):
   ones = n % 10
   tens = (n % 100) // 10
   hundreds = n // 100
   return ones + tens + hundreds
```