Generic counting function:

```
def some_counting_function(s):
  total = 0
  for pos in range(0, len(s), 1):
    if <test s[pos] for something>:
      total = total + 1
  return total
```

Generic filtering function:

```
def some_filtering_function(s):
  answer = ""
  for pos in range(0, len(s), 1):
    if <test s[pos] for something>:
      answer = answer + s[pos]
  return answer
```

Generic transforming function:

```
def some_transforming_function(s):
  answer = ""
  for pos in range(0, len(s), 1):
      newchar = <a new string probably based on s[pos] in some way>
      answer = answer + newchar
  return answer
```

Generic transforming with an if:

```
def some_transforming_function(s):
  answer = ""
  for pos in range(0, len(s), 1):
      if <some test on s[pos]>:
          answer = answer + <something>
      else:
          answer = answer + <something else>
  return answer
```

Transforming practice:

1. Write a function called change_nums that increments all numbers in a string by one:
   Example: change_nums("a1b2") returns "a2b3"

2. Write a function called encode that takes a string and encodes it using the simple cipher A=1, B=2, C=3, and so on. Make this work with uppercase and lowercase letters.
   Example: encode("abc") returns "1-2-3".

   Hint: use a variable letters = "abcdefgh…" and the find function.
   What is letters.find("a")? letters.find("b")?

3. Challenge (hard): write a decode function that decodes a string like “1-2-3” back into “abc”.