# Strings IV

**WARM UP:**

- Write a function called **`sum_digits`** that returns the sum of all the digits in a string.
  Example: **`sum_digits("abc123def5")`** returns 11

- (Harder) Write a function called **`strange`** that keeps all the digits in a string, but *only digits that are immediately preceded by a letter*. The first character in the string is guaranteed to be a letter.
  Example: **`strange("a16.3j4LM19")`** returns **`"141"`**

| | |
|---|---|
| `s.startswith(t)` | True if the string s begins with the string t. |
| `s.endswith(t)` | True if the string s ends with the string t. |

| | |
|---|---|
| `s.find(t)` | Returns the lowest index at which substring t is found inside s. |
| `s.find(t, p)` | Same as above, but starts searching at position p. |
| `s.replace(t, t2)` | Returns a copy of s with all occurrences of t replaced by t2. |

| | |
|---|---|
| `s.upper()` | Returns a copy of s with all letters converted to uppercase. |
| `s.lower()` | Returns a copy of s with all letters converted to lowercase. |

# Filtering

look at each character:

does this character match a pattern?

if yes, attach the character to the answer

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

# More sophisticated filtering

look at each character:

does this character match a pattern?

if yes, attach a different character to the answer

if no, attach yet another character to the answer

# More sophisticated filtering

Keep only uppercase characters within a string:

```
answer = ""
for pos in range(0, len(s)):
  if s[pos].isupper():
    answer = answer + s[pos]
```

# More sophisticated filtering

Replace uppercase characters within a string with stars:

```
answer = ""
for pos in range(0, len(s)):
  if s[pos].isupper():
    answer = answer + "*"
  else:
    answer = answer + s[pos]
```

- Write a function called change_nums that increments all numbers in a string by one:
  - Example: change_nums("a1b2") returns "a2b3"
- 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.
- 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")?
- Challenge (hard): write a decode function.