List and Strings
Problem

• A positive integer $n$ is perfect if the sum of its factors (excluding itself) is equal to $n$.

**Example**: 6 is perfect because $1 + 2 + 3 = 6$.

• Write a program that finds all perfect numbers between 1 and 10,000.
Operations that work on strings and lists

1. `x in s`, `x not in s`
2. `s + t`, `s*n`, `n*s`
3. `s[i]`, `s[i:j]`, `s[i:j:k]`
4. `len(s)`, `min(s)`, `max(s)`
5. `s.index(i)`, `s.count(i)`
## Accessing parts of lists and strings

$L = ["hi", 10, "bye", 100, -20, 123, 176, 3.45, 1, "it"]$

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;hi&quot;</td>
<td>10</td>
<td>&quot;bye&quot;</td>
<td>100</td>
<td>-20</td>
<td>123</td>
<td>176</td>
<td>3.45</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $L[2:5]$ is ["bye", 100, -20]
- $L[:2]$ is ["hi", 10]
- $L[4:4]$ is []
- $L[:len(L):2] = ["hi", "bye", -20, 176, 1]$
- $L[2:5][1] = 100$
- $L[1:5][:2] = [10, "bye"]$
The `len`, `min`, and `max` functions

- **`len(s)`** is the length of `s` (which may be a string or a list)
- **`min(s)` (or `max(s)`)** is the smallest (largest) element in `s`
  - If `s` is a list of numbers (integers and floats) these functions return the smallest (largest) number
  - If `s` is a list of strings, these functions return the lexicographically smallest (largest) string
  - If `s` is a string, these functions return the lexicographically smallest (largest) character in the string
  - If `s` is a list that contains a mixture of numeric and non-numeric objects, then the result is not specified by the language and you should not rely on such a result.
The “search” functions

- `s.index(i)` returns the index of the first occurrence of `i` in `s`
- `s.count(i)` returns the number of occurrences of `i` in `s`

```python
>>> L = [1, 3, 6] * 4
>>> L
[1, 3, 6, 1, 3, 6, 1, 3, 6, 1, 3, 6]
>>> L.index(3)
1
>>> L.count(3)
4
>>> L.index(0)
Traceback (most recent call last):
  File "<string>", line 1, in <fragment>
ValueError: 0 is not in list
>>> L.count(0)
0
```
Useful string operations

1. `str.find(s)`
2. `str.isalnum()`, `str.isalpha()`, `str.isdigit()`, `str.islower()`, `str.isupper()`, etc.
3. `str.upper()`, `str.lower()`
4. `str.split()`
5. `str.replace(old, new)`
The `find` function

```python
>>> s = "hello, how are you?"
>>> s.find("how")
7
>>> s.find("e")
1
>>> s.find("how", 2, 9)
-1
>>> s.find("how", 2, 10)
7
```
The `split` function

- `s.split()` returns a list obtained by splitting `s` into substrings obtained by deleting whitespaces.

Example:

```python
>>> s
'hello, how are you?'
>>> s.split()
['hello,', 'how', 'are', 'you?']
```
The `replace` function

- `s.replace(old, new)` returns a string obtained by replacing all occurrences of the old string by the new string

Example:

```python
>>> s
'hello, how are you?'
>>> s.replace(",", " ")
'hello  how are you?
'hello, who are you?'
```
Write a program that builds a dictionary of words by processing a given text.

- **Definition of a word:** Any contiguous sequence of characters that
  - starts at the beginning of a line or is immediately preceded by a whitespace or punctuation mark and
  - ends at the end of a line or is immediately followed by a whitespace or punctuation mark.