

# COMP10001 Foundations of Computing

## Semester 1, 2022

### Tutorial Solutions: Week 9

— VERSION: 1478, DATE: MAY 2, 2022 —

## Discussion

1. What is a “list comprehension”? How do we write one and how do they make our code simpler?

**A:** A list comprehension is a shortcut notation used to accomplish simple iteration tasks involving a collection (usually list, but also possibly a set or dictionary) in one line of code. List comprehensions are formed by wrapping a pair of brackets around `<expression>` `<for iteration statement>` `<optional if filter condition>`. The iteration statement will be run and for each iteration, the result of the expression will be added to the collection - which could be a list, set or dictionary. If a filter condition is included, the object will only be added if that condition evaluates to `True`. List comprehensions are useful to shorten repetitive, simple loops into readable single lines of code. They are especially useful for initialising lists. Avoid cramming too much into list comprehensions or nesting them inside each other: we want them to remain readable.

### Now try Exercise 1

2. Why do we use “files”? Could we use computers without them?

**A:** Files allow us to store data on a computer permanently: they will persist on storage media after a program is terminated, as opposed to internal data storage such as lists and dictionaries which exist in the computer's temporary memory and are erased when the program finishes. Files are also useful for storing large amounts of data in a structured way and sharing it with others. It would not be possible to use code for solving many big data problems without them. In fact, it would not be possible to write code without them!

3. What are the steps to reading and writing files?

**A:** In Python, to open a file we use the `open()` function, which takes two arguments: the file's filename as a string; and another string which represents the “mode” for access. Some common modes are `'r'` for reading; `'w'` for writing (erasing all file contents if the file exists initially); and `'a'` for appending to an already-existing file. `open()` returns a “file object” which is used to access the file. Some common ways to read a file are using the `file.read()` method to read a whole file, returning a string; `file.readline()` to read one line of the file, returning a string; and `file.readlines()` to read an entire file, returning a list with each row of the file split as a separate element in the list. To write, use the `file.write()` method to write a string to output. When finished with a file, be sure to close it with the `file.close()` method to prevent buffer errors.

4. What is a “csv” file and why is it useful for storing and manipulating data?

**A:** A csv (comma separated values) file is a text file stored in a particular format, similar to that of a spreadsheet. That is, as rows of data with individual values separated by a comma (,) and rows separated by a new line character `\n`. csv data is useful for storing statistics or measurement data because the structure of the file allows Python to read and process the data in its spreadsheet-like format. Python's `CSV` library is very useful for processing csv files.

### Now try Exercises 2 & 3

## Exercises

1. Evaluate the following list comprehensions. For each one, also write some python code to generate the same list without using a comprehension.

(a) `[(name, 0) for name in ("evelyn", "alex", "sam")]`

**A:**

```
[('evelyn', 0), ('alex', 0), ('sam', 0)]
```

```
my_list = []
for name in {"evelyn", "alex", "sam"}:
    my_list.append((name, 0))
```

(b) `[i**2 for i in range(5) if i % 2 == 1]`

A:

```
[1, 9]
```

```
my_list = []
for i in range(5):
    if i % 2 == 1:
        b.append(i*2)
```

(c) `"".join([letter.upper() for letter in "python"])`

A:

```
'PYTHON'
```

```
my_list = []
for letter in "python":
    my_list.append(letter.upper())
my_str = "".join(my_list)
```

2. Fill in the blanks in the program below which reads from `in.txt` and writes to `out.txt`.

```
outfile =  ("out.txt", "w")
with open("in.txt", ) as infile:
    line_no = 1
    for line in :
        outfile. (f"line:_{line_no},_length:_{len(line)}\n")
        line_no += 1
outfile.write("The_End")

```

A: (1) `open`  
(2) `'r'`  
(3) `infile.readlines()`  
(4) `write`  
(5) `outfile.close()`

3. “travel.csv” is a csv file containing data on how people get to work in different cities in Australia. “process.py” is a python program which processes this data. What information does the “process.py” attempt to find and print?

```
travel.csv
City,Train,Tram,Bus,Ferry,Car>Total
Melbourne,242969,55169,31937,783,1282997,1613855
Sydney,368572,3210,138340,9007,1206350,1725482
Adelaide,13715,4137,33673,211,390360,442102
Brisbane,62069,229,58228,3761,663353,787650
Perth,56417,223,37899,373,594571,689489
```

Data Source: Census of Population and Housing, 2016, TableBuilder

```
process.py
```

```
1 import csv
2
3 fp = open("travel.csv")
4 city = ''
5 curr_max = 0.0
6 for row in csv.DictReader(fp):
7     ferry = int(row["Ferry"])
8     total = int(row["Total"])
9     if ferry / total > curr_max:
10         city = row["City"]
11         curr_max = ferry / total
12 print(city)
```

**A:** “process.py” finds the city in which the highest proportion of its population uses the ferry to get to work. It will print *Sydney*. To customise this program, we could change line 7 to fetch a different column, giving it a new header string such as “Train” or “Car”.  
As an exercise, try writing a program which prints the proportion of transport mode used in each city.

## Problems

1. Let’s head back to worksheet 8 and solve “For a While ...” again! **Using a list comprehension**, rewrite the function `allnum` that takes a list of strings, and returns a list of those that exclusively contain digits. `allnum(['3', '-4', '5', '3.1416', '0xffff', 'blerg!'])` should return `['3', '5']`.

**A:**

```
def allnum(strlist):  
    return [curr_str for curr_str in strlist if curr_str.isdigit()]
```

2. Write a function which takes a filename as a string and opens the file, printing statistics for every line in the file: the line number, number of words, number of characters. Remember to close the file when you’re done!

**A:**

```
def statistics(filename):  
    f = open(filename, 'r')  
    line_no = 1  
    print("Line_number:_num_words,_num_characters")  
    for line in f.readlines():  
        print(f"{line_no}:_{len(line.split())}_{len(line)}")  
        line_no += 1  
    f.close()
```