Week 10 Tutorial

COMP10001 – Foundations of Computing

Semester 1, 2025

Clement Chau

- Files I/O
- CSV Files

Revision: Files

type <str>
"r": read mode
"w": write mode
"a": append mode

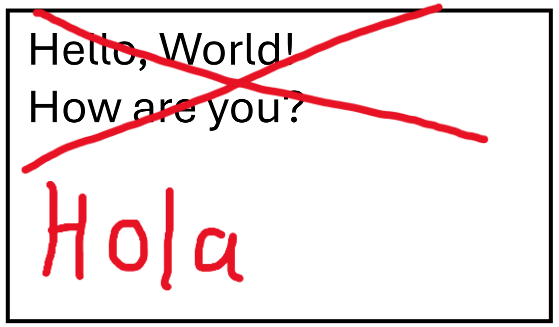
file pointer

```
fp = open(file_name, mode)
```

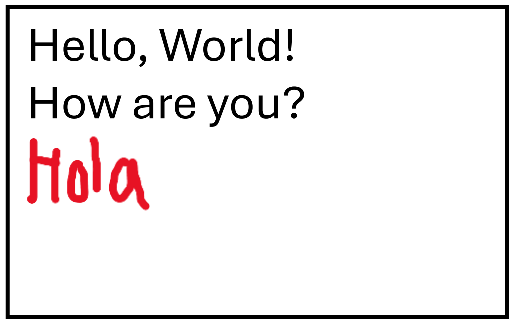
type <str>
e.g. "story.txt", "data.csv"

Revision: Files modes

- "**r**"ead mode
  - Assumes that the file exists, otherwise returns a FileNotFound error.
- "**w**"rite mode
  - If file exists, it **erases** all its contents (even before writing to it!).
  - If file does not exist, it implicitly creates a new file first.
- "**a**"ppend mode
  - If file exists, writing to it will append to the end of the file.



write mode



append mode

Revision: Files methods (read)

- fp = open("hello.txt", "r")
- print(fp.**read()**)
- **"Hello, World!\nHow are you?\nI'm good!"**

newline character

- fp.**close()**

hello.txt

```
Hello, World!
How are you?
I'm good!
```

Revision: Files methods (readline)

- fp = open("hello.txt", "r")
- print(fp.**readline()**)
- **"Hello, World!"**
- print(fp.**readline()**)
- **"How are you?"**
- fp.**close()**

hello.txt

```
Hello, World!
How are you?
I'm good!
```

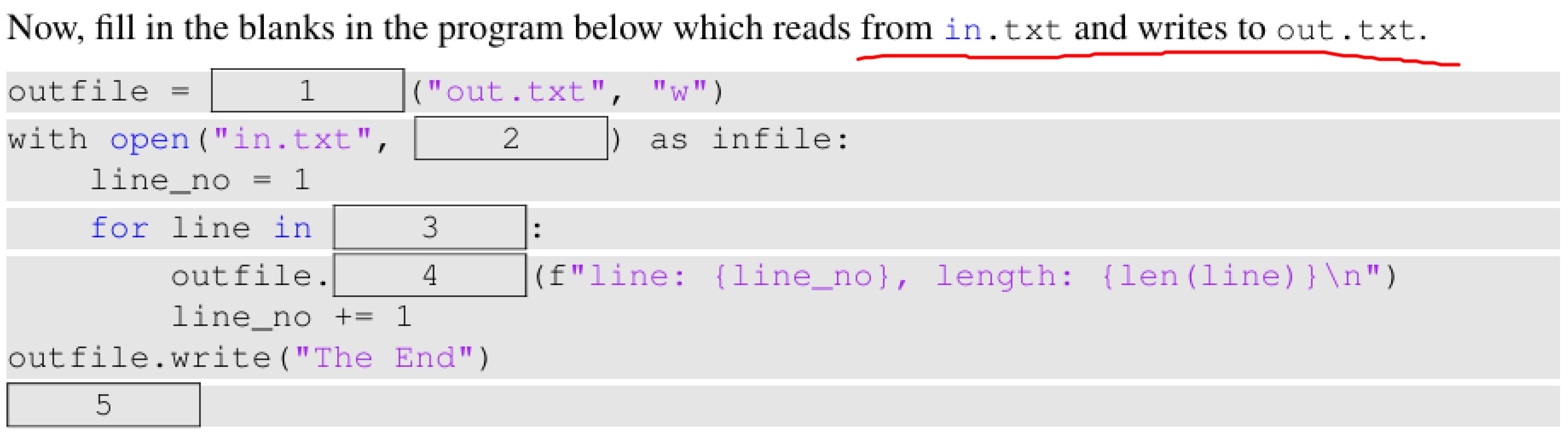
Revision: Files methods (readlines)

- fp = open("hello.txt", "r")
- print(fp.**readlines()**)
- **["Hello, World!",**
  "How are you?",
  "I'm good!"]
- fp.**close()**

hello.txt

```
Hello, World!
How are you?
I'm good!
```

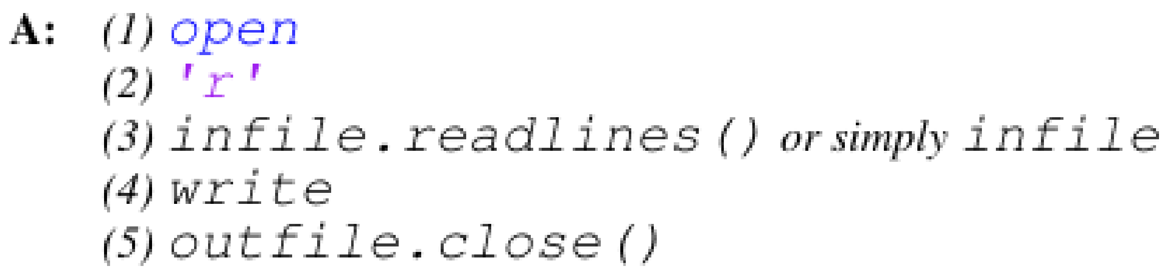
Exercise 1 / 1



Now, fill in the blanks in the program below which reads from `in.txt` and writes to `out.txt`.

```
outfile = [    1    ]("out.txt", "w")
with open("in.txt", [    2    ]) as infile:
    line_no = 1
    for line in [    3    ]:
        outfile.[    4    ](f"line: {line_no}, length: {len(line)}\n")
        line_no += 1
outfile.write("The End")
[    5    ]
```

Answer:

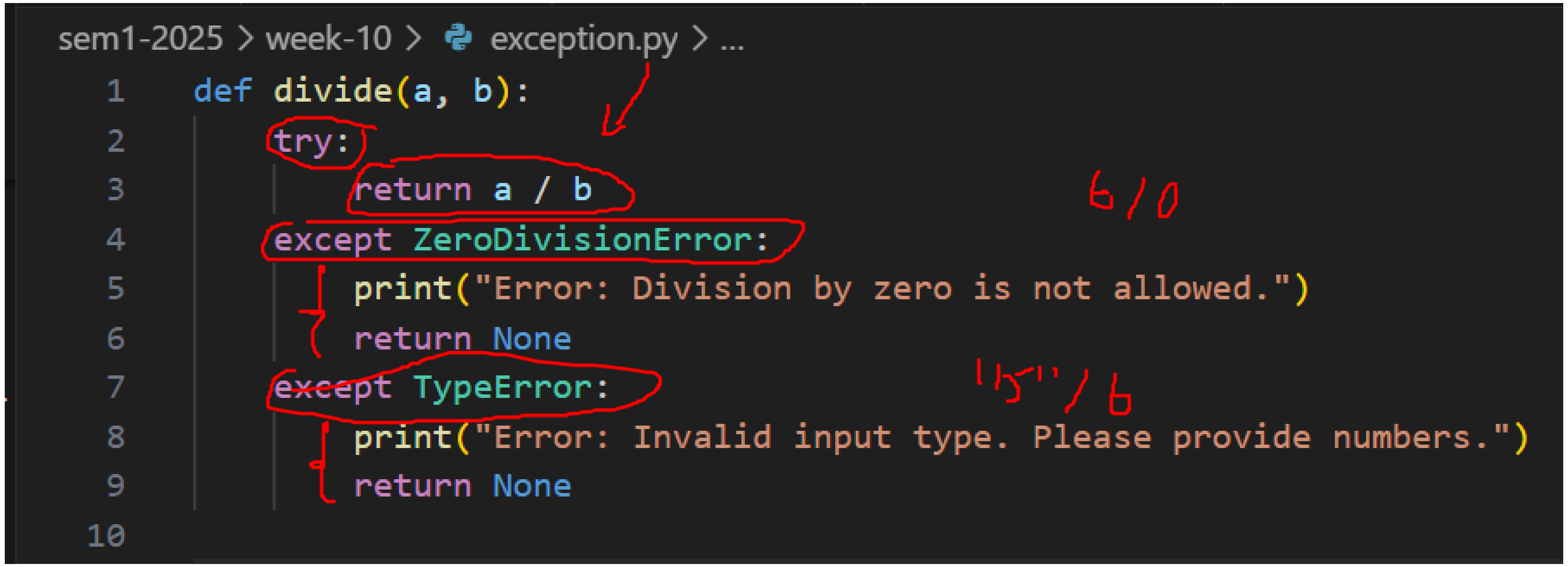


A: *(1)* `open`
(2) `'r'`
(3) `infile.readlines()` *or simply* `infile`
(4) `write`
(5) `outfile.close()`

Programming Problems

Revision: Exceptions

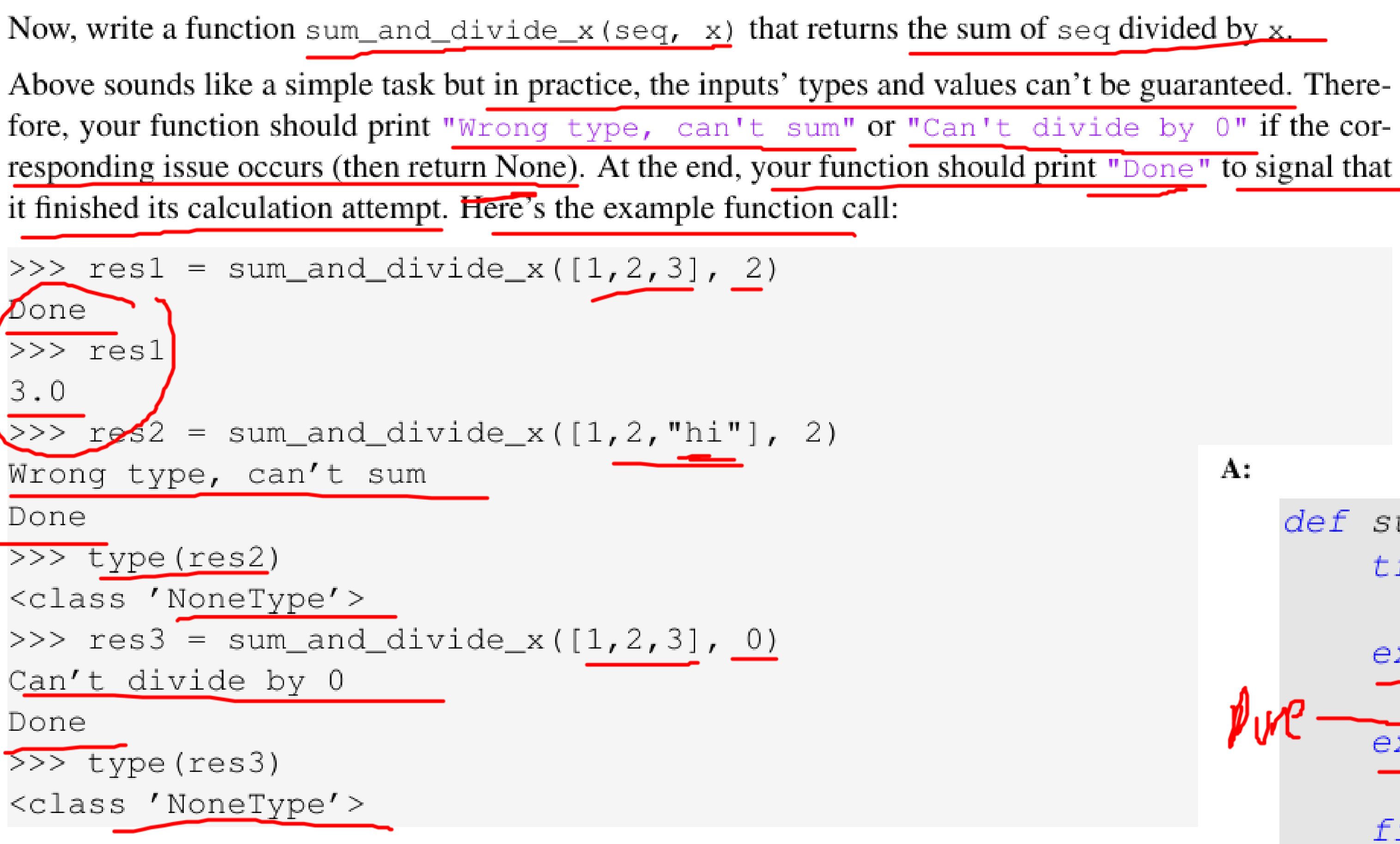
```
try:
    # code block where an exception might occur
except ExceptionType:
    # code block to handle the exception
finally:
    # code block that will always execute, regardless of
    # whether an exception was raised or not
```



sem1-2025 > week-10 > exception.py > ...

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed.")
        return None
    except TypeError:
        print("Error: Invalid input type. Please provide numbers.")
        return None
```

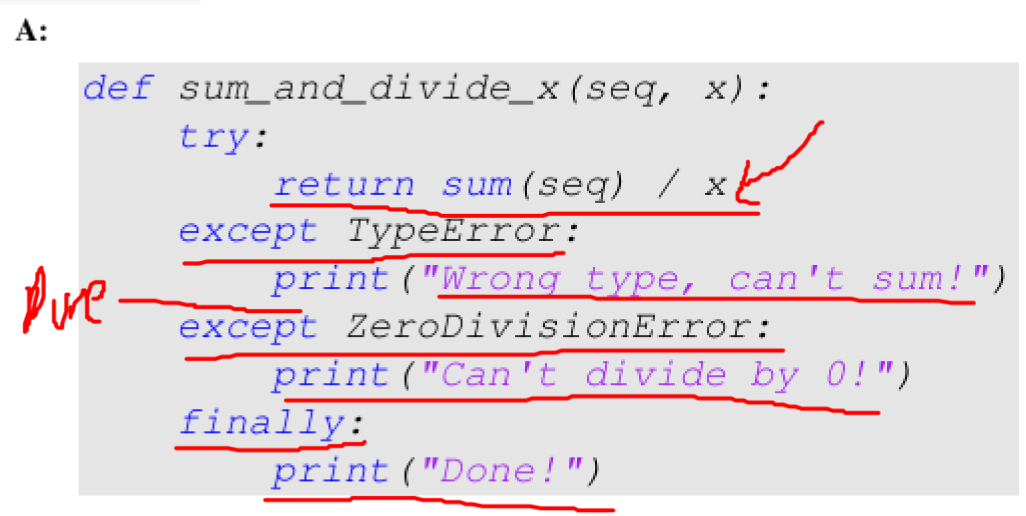
Problem 1 / 3


Now, write a function `sum_and_divide_x(seq, x)` that returns the sum of `seq` divided by `x`.

Above sounds like a simple task but in practice, the inputs' types and values can't be guaranteed. Therefore, your function should print `"Wrong type, can't sum"` or `"Can't divide by 0"` if the corresponding issue occurs (then return None). At the end, your function should print `"Done"` to signal that it finished its calculation attempt. Here's the example function call:

```
>>> res1 = sum_and_divide_x([1,2,3], 2)
Done
>>> res1
3.0
>>> res2 = sum_and_divide_x([1,2,"hi"], 2)
Wrong type, can't sum
Done
>>> type(res2)
<class 'NoneType'>
>>> res3 = sum_and_divide_x([1,2,3], 0)
Can't divide by 0
Done
>>> type(res3)
<class 'NoneType'>
```

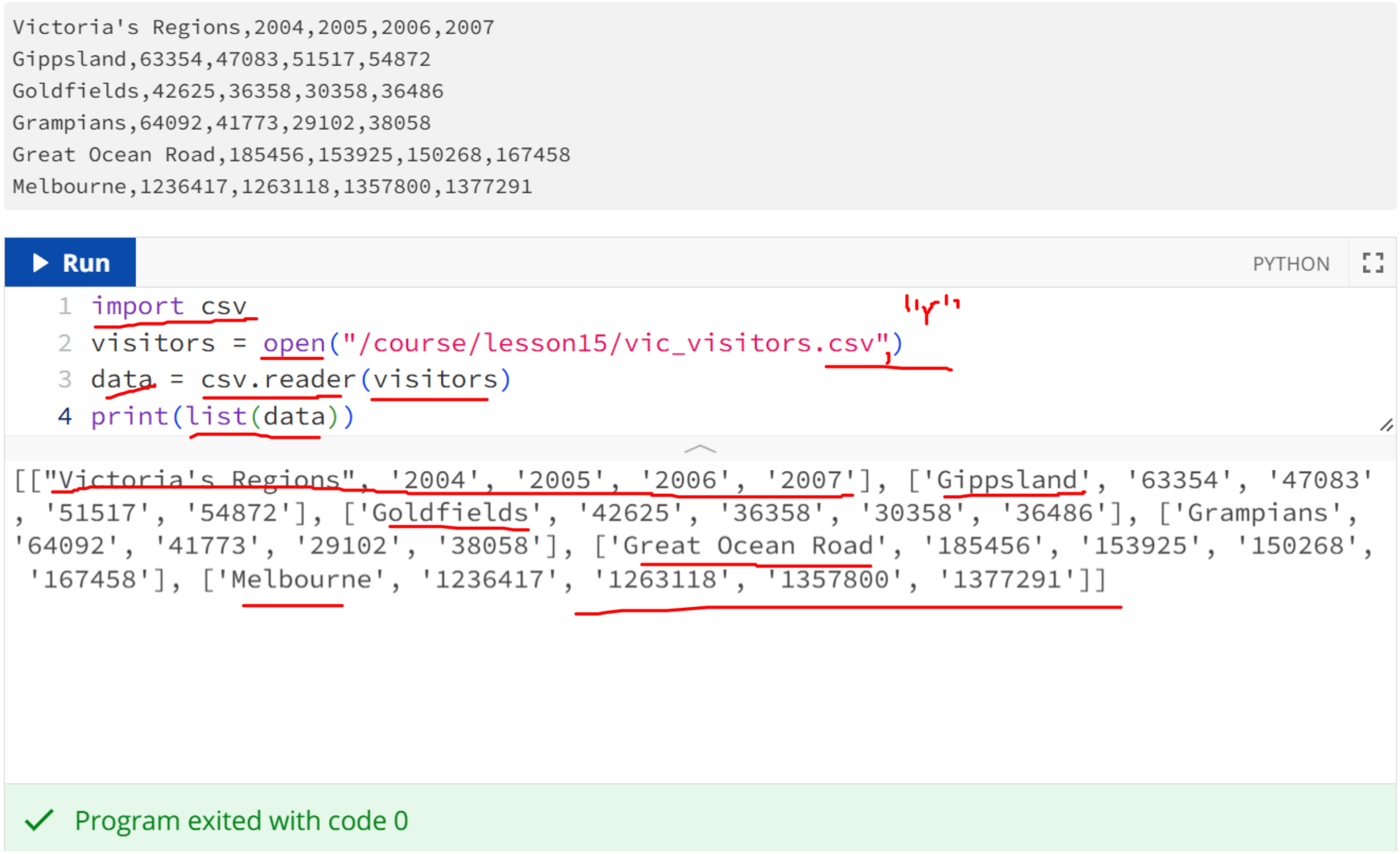
Answer:


A:

```
def sum_and_divide_x(seq, x):
    try:
        return sum(seq) / x
    except TypeError:
        print("Wrong type, can't sum!")
    except ZeroDivisionError:
        print("Can't divide by 0!")
    finally:
        print("Done!")
```

Pure

Revision: CSV Files Methods (reader)



```
Victoria's Regions,2004,2005,2006,2007
Gippsland,63354,47083,51517,54872
Goldfields,42625,36358,30358,36486
Grampians,64092,41773,29102,38058
Great Ocean Road,185456,153925,150268,167458
Melbourne,1236417,1263118,1357800,1377291
```

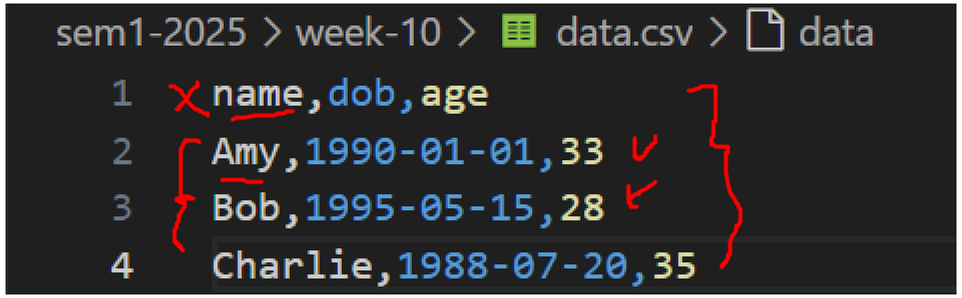
Run PYTHON

```
import csv
                                                   'r'
visitors = open("/course/lesson15/vic_visitors.csv",)
data = csv.reader(visitors)
print(list(data))
```

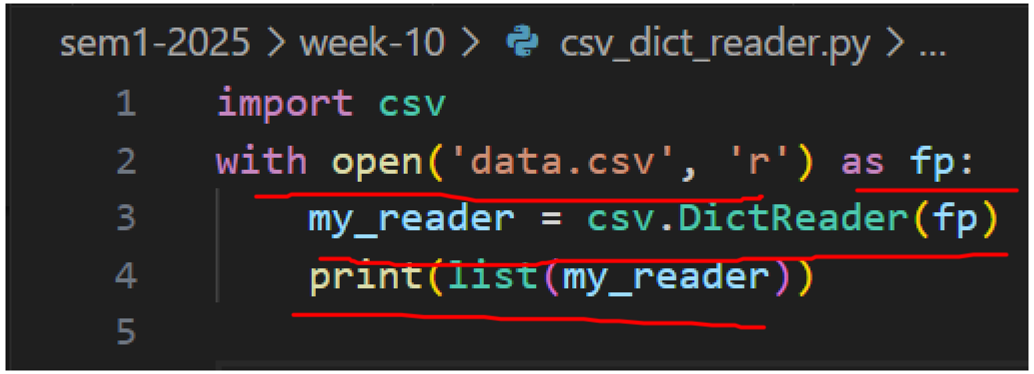
```
[["Victoria's Regions", '2004', '2005', '2006', '2007'], ['Gippsland', '63354', '47083'
, '51517', '54872'], ['Goldfields', '42625', '36358', '30358', '36486'], ['Grampians',
'64092', '41773', '29102', '38058'], ['Great Ocean Road', '185456', '153925', '150268',
 '167458'], ['Melbourne', '1236417', '1263118', '1357800', '1377291']]
```

✓ Program exited with code 0

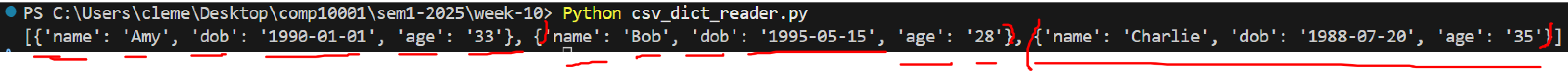
Revision: CSV Files Methods (DictReader)


sem1-2025 > week-10 > data.csv > data

```
name,dob,age
Amy,1990-01-01,33
Bob,1995-05-15,28
Charlie,1988-07-20,35
```

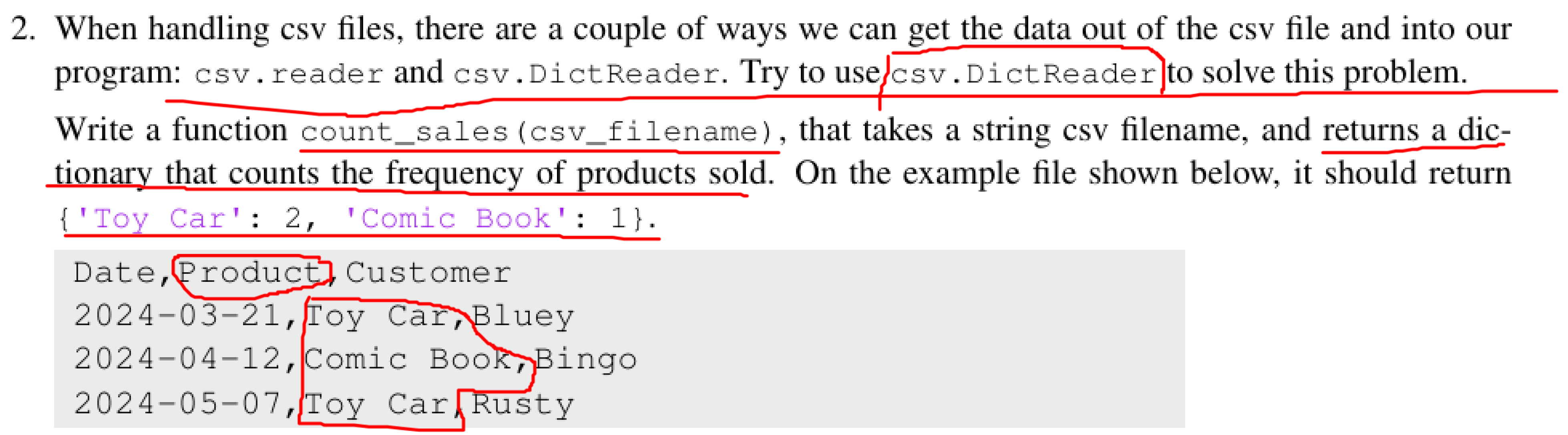

sem1-2025 > week-10 > csv_dict_reader.py > ...

```python
import csv
with open('data.csv', 'r') as fp:
    my_reader = csv.DictReader(fp)
    print(list(my_reader))
```



```
PS C:\Users\cleme\Desktop\comp10001\sem1-2025\week-10> Python csv_dict_reader.py
[{'name': 'Amy', 'dob': '1990-01-01', 'age': '33'}, {'name': 'Bob', 'dob': '1995-05-15', 'age': '28'}, {'name': 'Charlie', 'dob': '1988-07-20', 'age': '35'}]
```

Problem 2 / 3

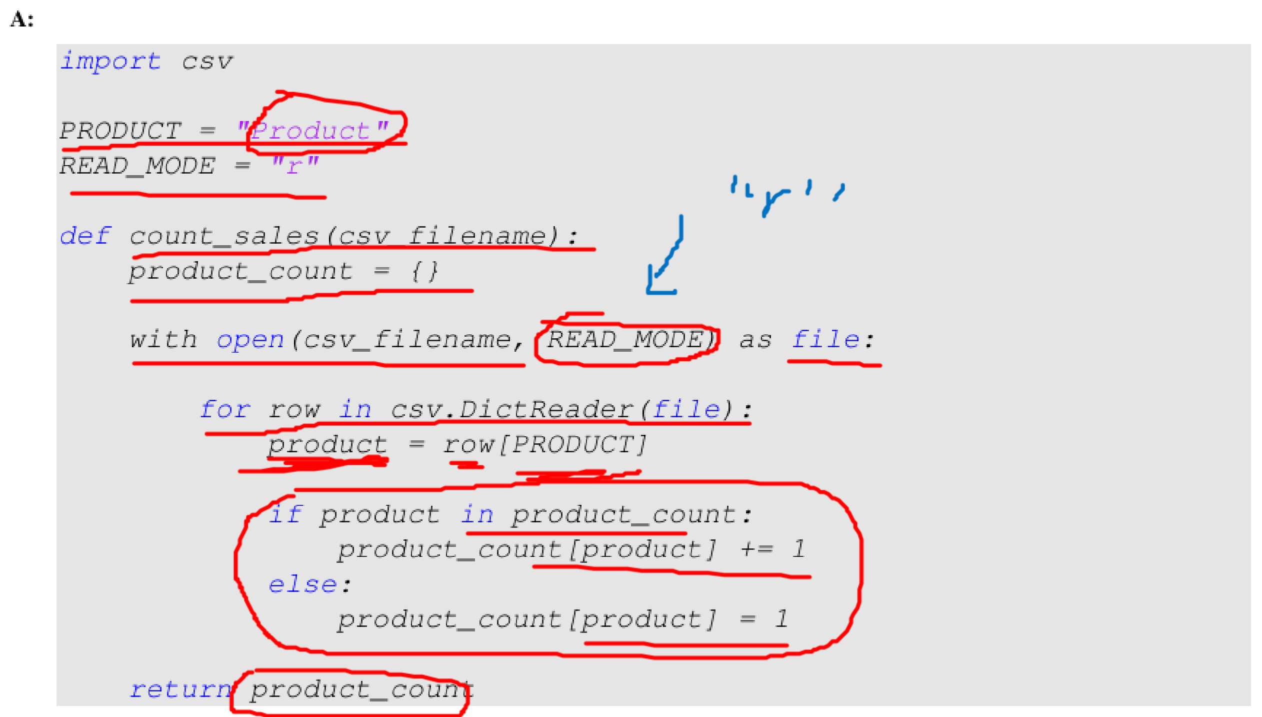


2. When handling csv files, there are a couple of ways we can get the data out of the csv file and into our program: `csv.reader` and `csv.DictReader`. Try to use `csv.DictReader` to solve this problem.

   Write a function `count_sales(csv_filename)`, that takes a string csv filename, and returns a dictionary that counts the frequency of products sold. On the example file shown below, it should return `{'Toy Car': 2, 'Comic Book': 1}`.

```
Date,Product,Customer
2024-03-21,Toy Car,Bluey
2024-04-12,Comic Book,Bingo
2024-05-07,Toy Car,Rusty
```

Answer:



A:

```
import csv

PRODUCT = "Product"
READ_MODE = "r"

def count_sales(csv_filename):
    product_count = {}

    with open(csv_filename, READ_MODE) as file:

        for row in csv.DictReader(file):
            product = row[PRODUCT]

            if product in product_count:
                product_count[product] += 1
            else:
                product_count[product] = 1

    return product_count
```

'r'

Problem 3 / 3

3. **Challenge:** You've found a secret message:

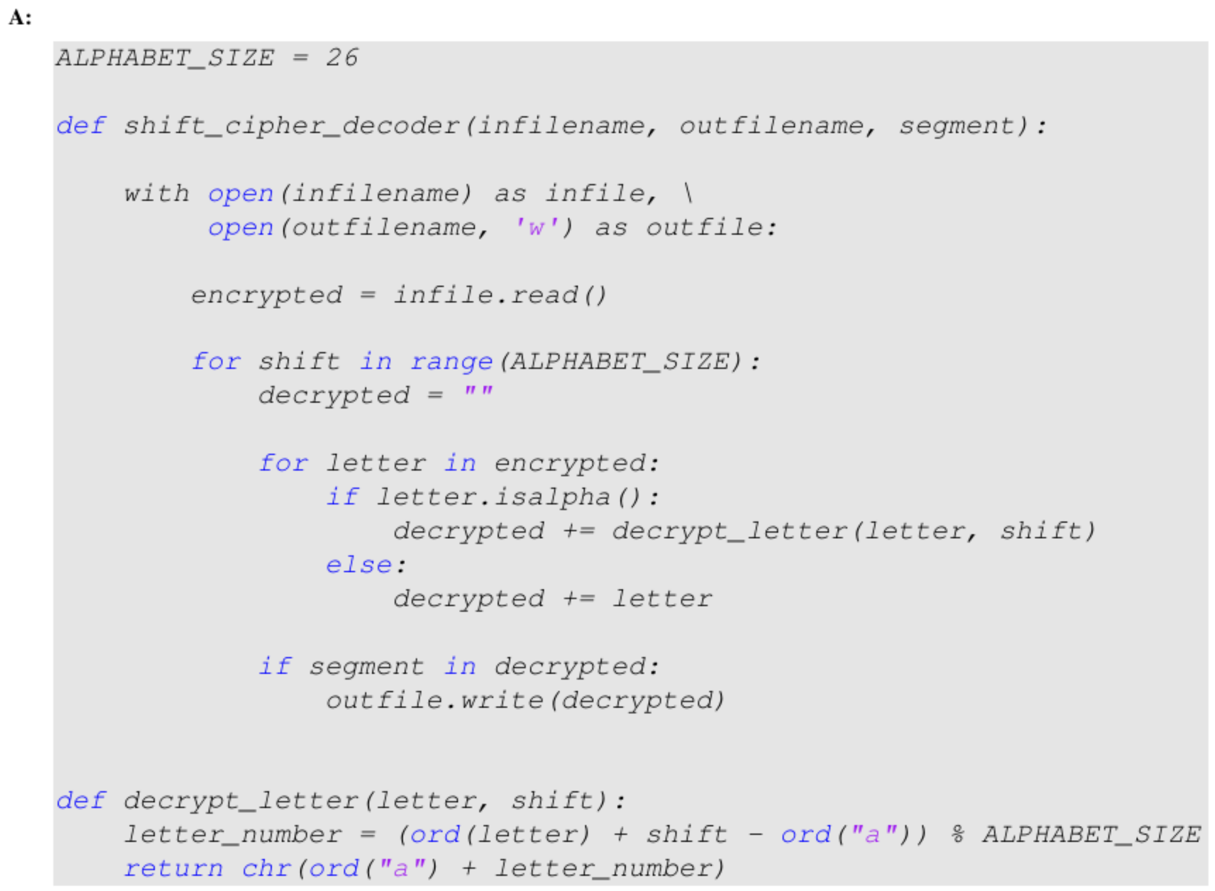
```
secret_message.txt
```

```
erkbvl ur kbvd tlmexr:
gxoxk zhggt zbox rhn ni
gxoxk zhggt exm rhn whpg
gxoxk zhggt kng tkhngw tgw wxlxkm rhn
gxoxk zhggt ftdx rhn vkr
gxoxk zhggt ltr zhhwurx
gxoxk zhggt mxee t ebx tgw ankm rhn
```

All that you know about the message is that it is encrypted by a basic shift cipher (also known as a Caesar cipher, where each letter is shifted by some constant number of places in the alphabet), any alphabetic character in the message is lowercase, and that it contains the string segment `'desert'`.

Write a function to decrypt the message that takes an `infilename`, `outfilename` and `segment` (all strings, and you can assume that all files exist). You can use a brute-force approach (try all possible values) to guess the number to shift by. You might find the functions `ord(character)` and `chr(number)` useful!

Answering to Problem 3:



A:

```
ALPHABET_SIZE = 26

def shift_cipher_decoder(infilename, outfilename, segment):

    with open(infilename) as infile, \
         open(outfilename, 'w') as outfile:

        encrypted = infile.read()

        for shift in range(ALPHABET_SIZE):
            decrypted = ""

            for letter in encrypted:
                if letter.isalpha():
                    decrypted += decrypt_letter(letter, shift)
                else:
                    decrypted += letter

            if segment in decrypted:
                outfile.write(decrypted)


def decrypt_letter(letter, shift):
    letter_number = (ord(letter) + shift - ord("a")) % ALPHABET_SIZE
    return chr(ord("a") + letter_number)
```

Revision Problems

(Not necessarily related to this week's topic, but more so for general exam practice)

Revision Problem 1 / 2

1. Write a function that takes a lowercase string as input and prints the frequency of each vowel in the string. The printed results should be in alphabetical order. `vowel_counts('i love python')` should print:

```
e 1
i 1
o 2
```

Answer:

A:

```python
from collections import defaultdict as dd
VOWELS = 'aeiou'

def vowel_counts(text):
    vowel_counts = dd(int)
    for letter in text:
        if letter in VOWELS:
            vowel_counts[letter] += 1
    for vowel, count in sorted(vowel_counts.items()):
        print(vowel, count)
```

'aeiou'

Revision Problem 2 / 2

2. Write a function which takes two lists of integers and returns the average of the numbers which they both have in common. `in_common_average([1, 2, 3, 4, 5], [0, 2, 4, 6])` should return `3.0`

$$\frac{2+4}{2} = 3.0$$

Answer:

A:

```python
def in_common_average(list1, list2):
    common = set(list1) & set(list2)
    return sum(common) / len(common)
```

Independent Work

- **NO Ed Worksheets due next week.**
    - Please focus on your **Project 2**. This is **due Friday, May 23rd, 6pm**.
        - Project 2 is (considerably) more difficult than Project 1. **Start early**.
    - We are hoping to release Project 1 marks by the end of the week
        - I have sent an email to everyone regarding **general feedback**. I highly recommend reading it and incorporating them into Project 2.
        - If you have any questions regarding your marks for Project 1, my email is **clement.chau@unimelb.edu.au**.
- **Raise your hand** if you have any questions!

Scan here for annotated slides

Project 1 **<u>General</u>** Feedback