L7 "Homework"

Alek Francescangeli

June 13, 2023

Contents

1 Problem 1

1.1 Run-Down

Front-end: on the front end we are using html/css/javascript with libraries such as Vue, Papaparse, and highcharts. We will parse the csv into json, we will send a POST request to the back-end in order to get information/analysis on the data.

Back-end: on the backend we are using flask with pandas in order to look for data that starts with numbers 1-9 in order to analyze its conformity to Benford's Law.

To run, please install requirements in the 'requirements.txt' file and then run base.py

To build a docker image to run from please run (from root directory):

```
docker build -t benfordapp .
```

Docker Image:
docker pull antisage9/benfordapp:latest

1.2 How I would further develop this

-Find out more about who would be using this app and in what capacity. This would affect what kind of usability features we would want to add to the application.

-Add significantly more testing, presently only the get conformity function in the base python file has unit tests, adding more tests to prove we are getting the proper

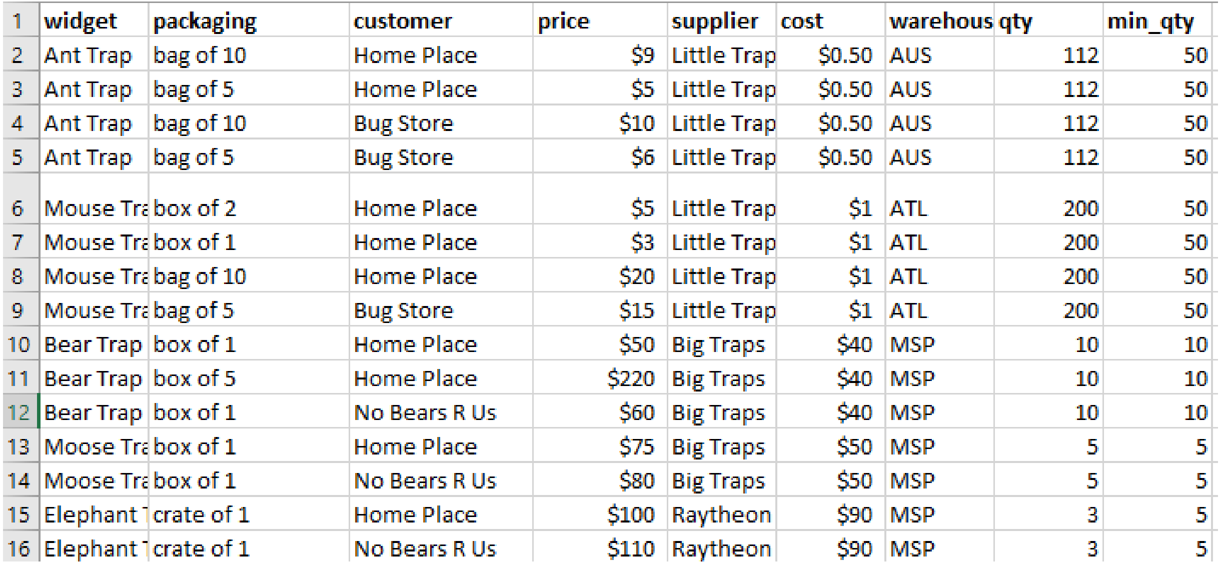
-Set up a pipeline for future improvements, Gitlab or Jenkins could be used for this.

2 Problem 2

Problem: Create a basic system description and document a normalized schema from the attached widgets (widgets.tsv) text file. Include 1) what you think this system would do 2) what you feel would be a reasonable database structure for the data and a reasonable architecture for the system 3) any questions or concerns you have regarding this dataset/system that might need to be answered before establishing an ideal database/solution for such a system. It's a very open-ended problem.

2.1 What This System Will Do

The following was given in the widgets.tsv file:



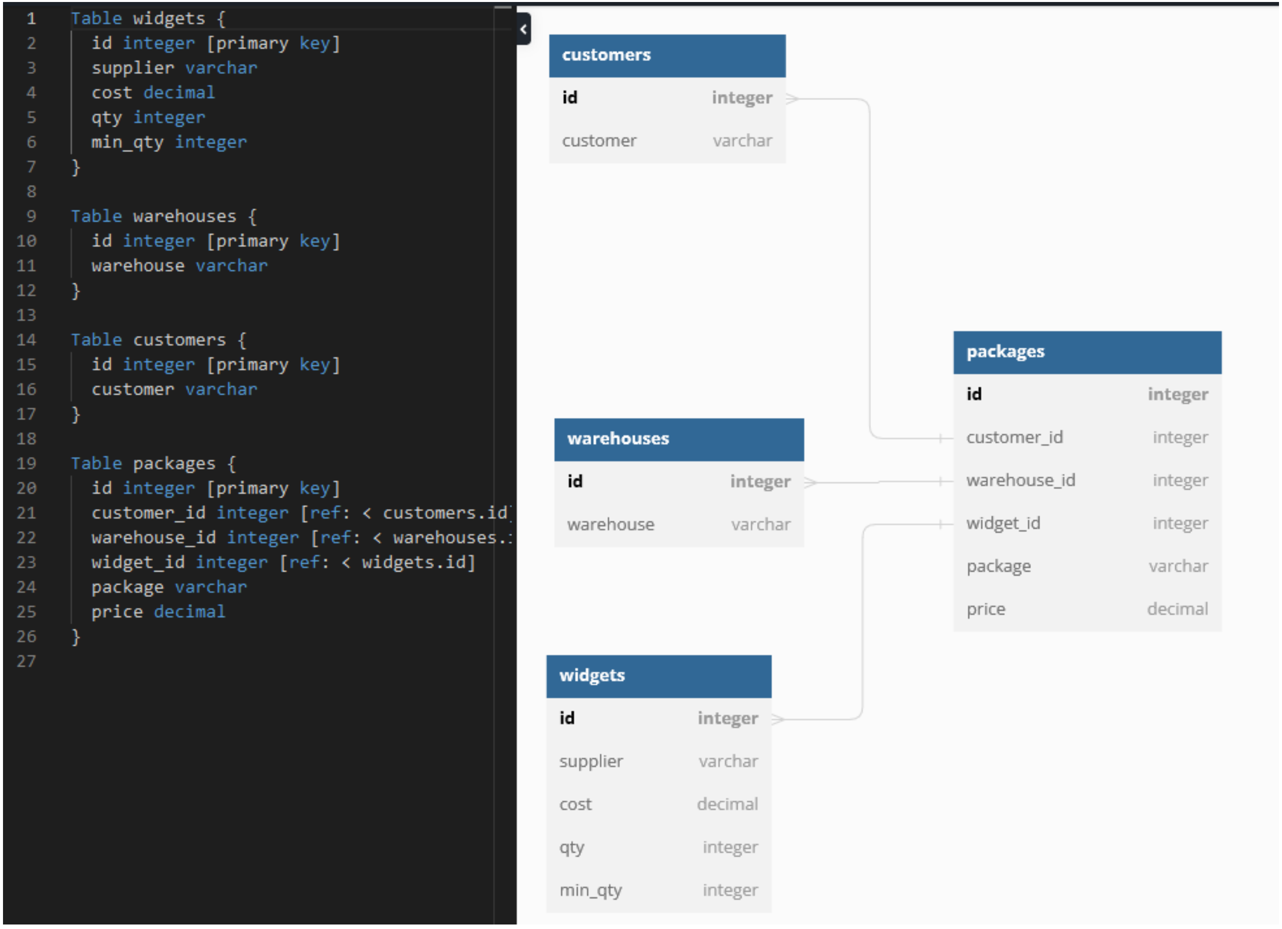
1	widget	packaging	customer	price	supplier	cost	warehous	qty	min_qty
2	Ant Trap	bag of 10	Home Place	$9	Little Trap	$0.50	AUS	112	50
3	Ant Trap	bag of 5	Home Place	$5	Little Trap	$0.50	AUS	112	50
4	Ant Trap	bag of 10	Bug Store	$10	Little Trap	$0.50	AUS	112	50
5	Ant Trap	bag of 5	Bug Store	$6	Little Trap	$0.50	AUS	112	50
6	Mouse Tra	box of 2	Home Place	$5	Little Trap	$1	ATL	200	50
7	Mouse Tra	box of 1	Home Place	$3	Little Trap	$1	ATL	200	50
8	Mouse Tra	bag of 10	Home Place	$20	Little Trap	$1	ATL	200	50
9	Mouse Tra	bag of 5	Bug Store	$15	Little Trap	$1	ATL	200	50
10	Bear Trap	box of 1	Home Place	$50	Big Traps	$40	MSP	10	10
11	Bear Trap	box of 5	Home Place	$220	Big Traps	$40	MSP	10	10
12	Bear Trap	box of 1	No Bears R Us	$60	Big Traps	$40	MSP	10	10
13	Moose Tra	box of 1	Home Place	$75	Big Traps	$50	MSP	5	5
14	Moose Tra	box of 1	No Bears R Us	$80	Big Traps	$50	MSP	5	5
15	Elephant T	crate of 1	Home Place	$100	Raytheon	$90	MSP	3	5
16	Elephant T	crate of 1	No Bears R Us	$110	Raytheon	$90	MSP	3	5

From this file we can most likely assume that it will be used to keep track of sales and inventory of various widgets to different customers. It could also be used to enter in new packaging sales.

2.2 Database Structure

I think there are four major tables that we would have to keep track of in this schema, **Customer**, **Warehouse**, and **Widget**. After more thought I decided a 4th **Package** table would be the best way to do this as it could hold distinct package information that was not unique to any of the previous 3 mentioned tables. Each table has a one-to-many relationship to the package table giving it all the pertinent information.

Here is the schema for this design, this was created using dbdiagram.io (link here):



```
Table widgets {
  id integer [primary key]
  supplier varchar
  cost decimal
  qty integer
  min_qty integer
}

Table warehouses {
  id integer [primary key]
  warehouse varchar
}

Table customers {
  id integer [primary key]
  customer varchar
}

Table packages {
  id integer [primary key]
  customer_id integer [ref: < customers.id
  warehouse_id integer [ref: < warehouses.:
  widget_id integer [ref: < widgets.id]
  package varchar
  price decimal
}
```

customers	
id	integer
customer	varchar

packages	
id	integer
customer_id	integer
warehouse_id	integer
widget_id	integer
package	varchar
price	decimal

warehouses	
id	integer
warehouse	varchar

widgets	
id	integer
supplier	varchar
cost	decimal
qty	integer
min_qty	integer

2.3 Architecture

This is a simple architecture that could be used for this application:

User Interface/Client Side: A web-based interface could be used to display records to the user and allow them to enter records. The technology for this would be HTML/CSS/JavaScript

Server-Side: Spring boot would be used to handle incoming API requests that come in. Data would be retrieved/updated using commands like GET/POST/PUT

Database: The database could be implemented with MySQL server

2.4 Questions/Concerns

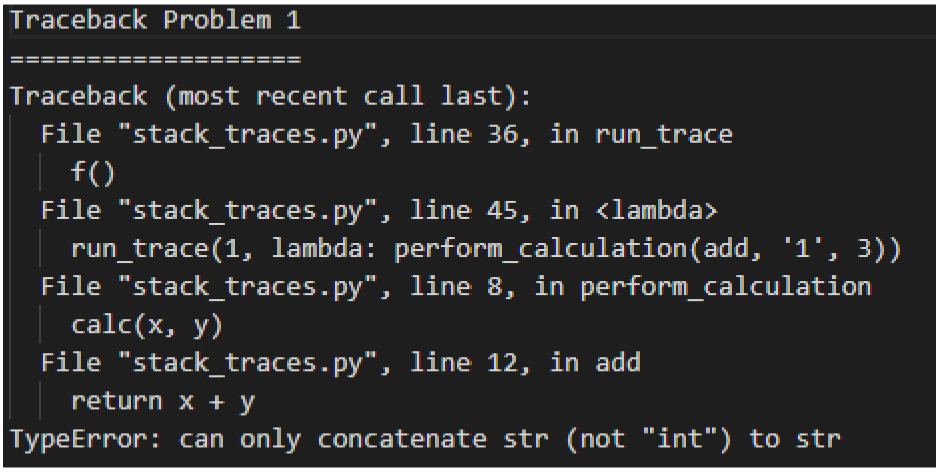
1.) Purpose of application
2.) Number of users
3.) Amount of data we will have to handle
4.) If a widget can be supplied by multiple suppliers
5.) Timeline this needs to be completed in

3 Problem 3

Problem: See the "Python Stack Traces" attachment which lists several python stack traces. Your task is to examine the stack traces and provide a brief response for each one that summarizes what the problem or likely problem is, and the first line of code you would jump to in your code editor given the trace.

Assumption: in these problems I opted to not change the input for the problem and went for trying to change the function to accept the given input (or at least handle it). Was not sure if this was the intended solution. Most of these problems seem like they could be solved easily by just changing the input to be the expected types.

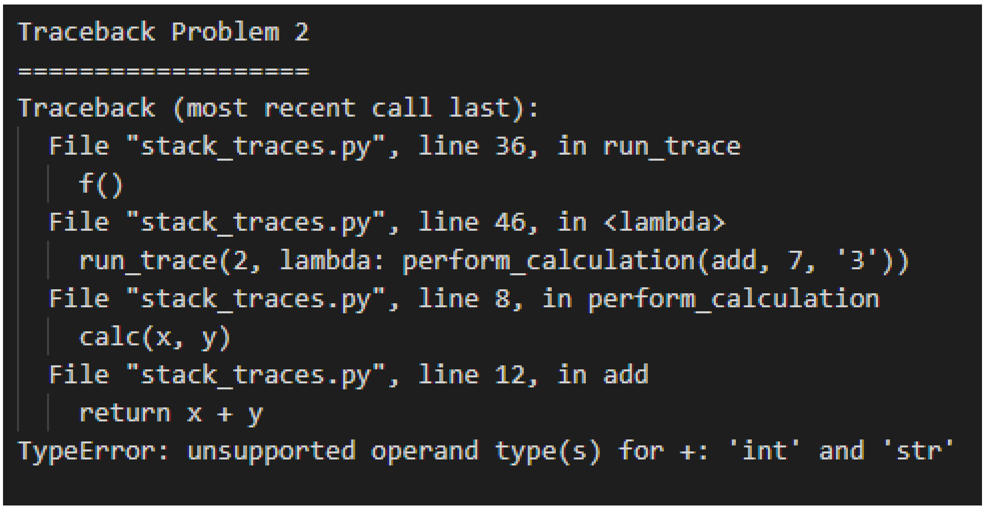
3.1 Trace 1



```
Traceback Problem 1
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 45, in <lambda>
    run_trace(1, lambda: perform_calculation(add, '1', 3))
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 12, in add
    return x + y
TypeError: can only concatenate str (not "int") to str
```

The likely problem here is that we are trying to concat a string and a number using "+", likely the '1' and 3. I would try to make sure that both inputs are compatible types in line 8 (perform calculations function)

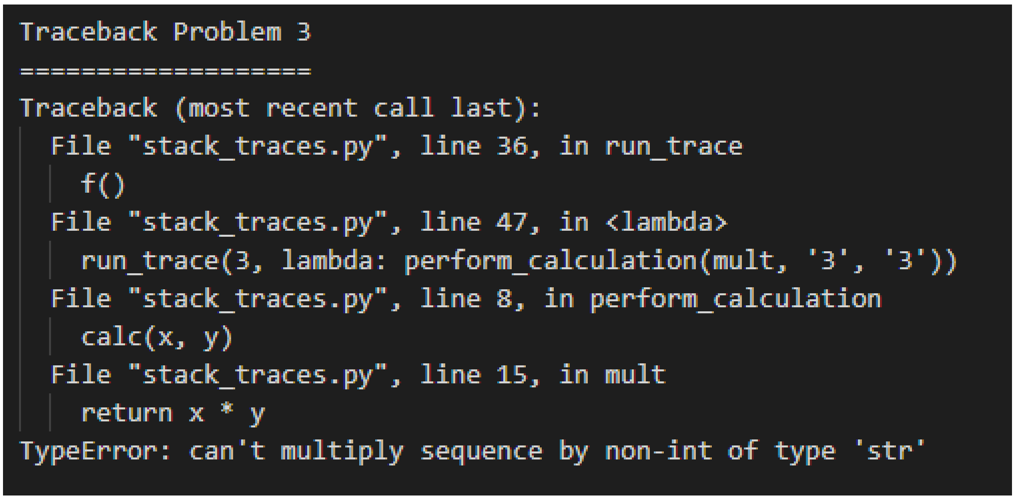
3.2 Trace 2



```
Traceback Problem 2
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 46, in <lambda>
    run_trace(2, lambda: perform_calculation(add, 7, '3'))
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 12, in add
    return x + y
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Same issue as the last trace, except we are trying to concat a int with a string instead of the other way around. I would jump to line 8 again to try to resolve this.

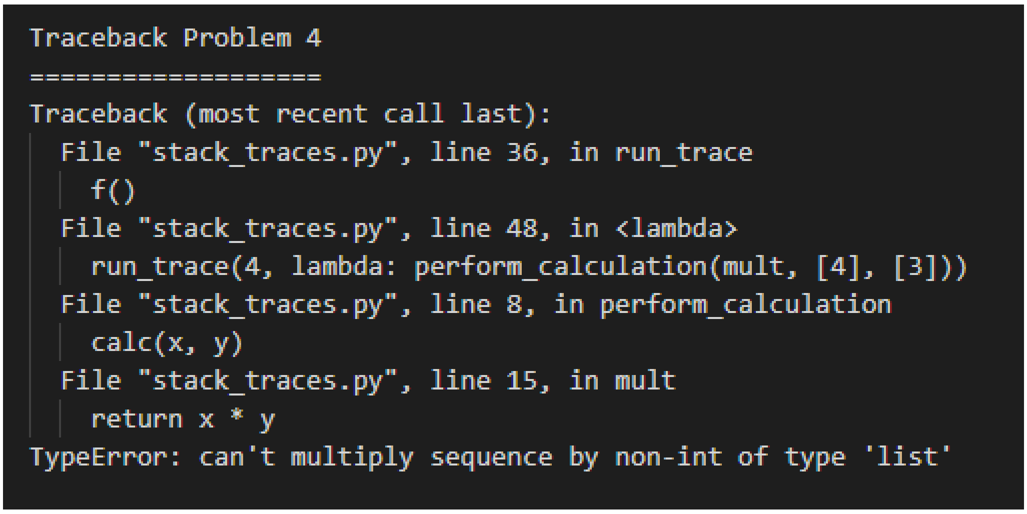
3.3 Trace 3



```
Traceback Problem 3
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 47, in <lambda>
    run_trace(3, lambda: perform_calculation(mult, '3', '3'))
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 15, in mult
    return x * y
TypeError: can't multiply sequence by non-int of type 'str'
```

The issue here is that we are trying to multiply two strings using ’*’...you can’t do that. I would look at line 8 to make sure we are checking input before executing.

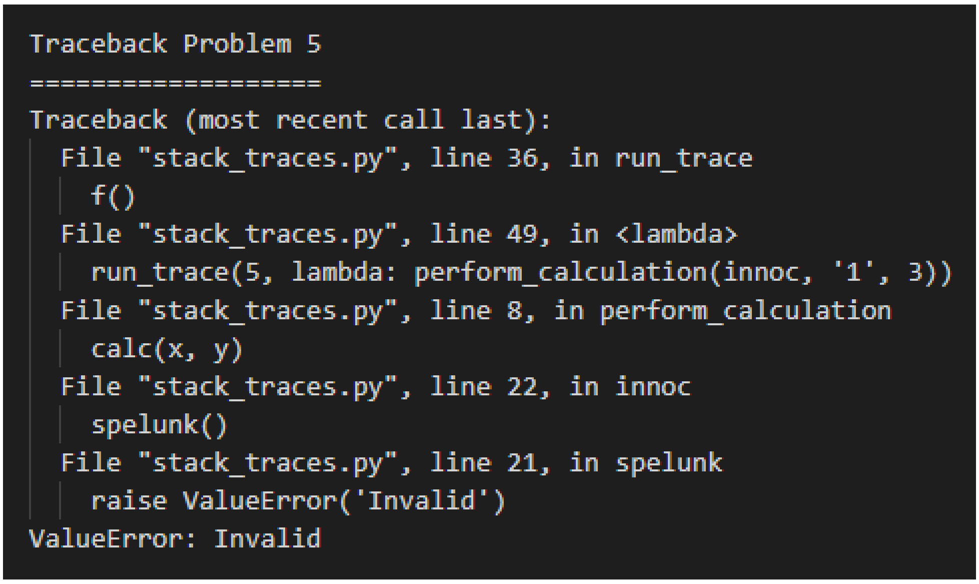
3.4 Trace 4



```
Traceback Problem 4
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 48, in <lambda>
    run_trace(4, lambda: perform_calculation(mult, [4], [3]))
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 15, in mult
    return x * y
TypeError: can't multiply sequence by non-int of type 'list'
```

The issue here is that we are trying to multiply two lists using ’*’...you can’t do that. I would look at line 8 to make sure we are checking input before executing.

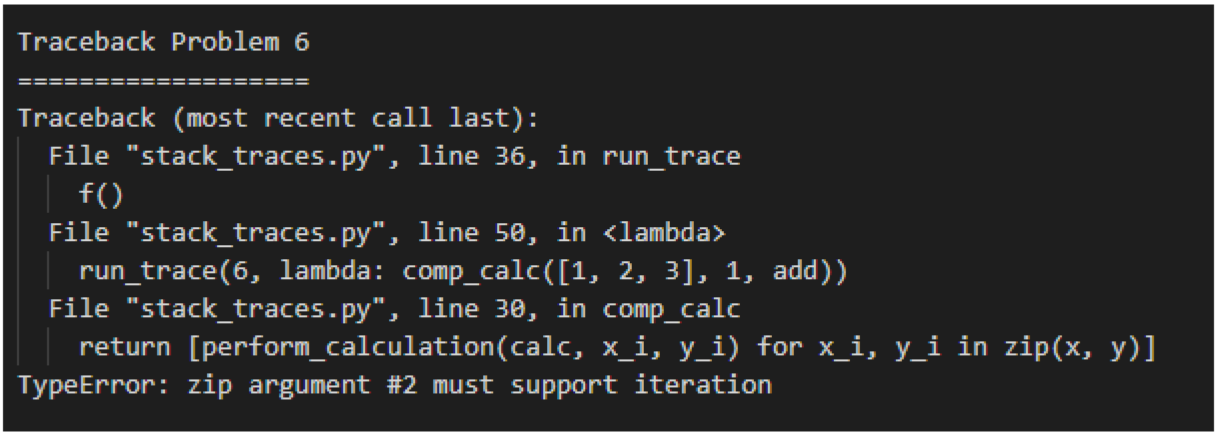
3.5 Trace 5



```
Traceback Problem 5
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 49, in <lambda>
    run_trace(5, lambda: perform_calculation(innoc, '1', 3))
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 22, in innoc
    spelunk()
  File "stack_traces.py", line 21, in spelunk
    raise ValueError('Invalid')
ValueError: Invalid
```

The issue here is that we are raising a error in ”spelunk” (line 21) but never handling it. I would start by looking at line 22 to see how innoc would possibly handle the error.

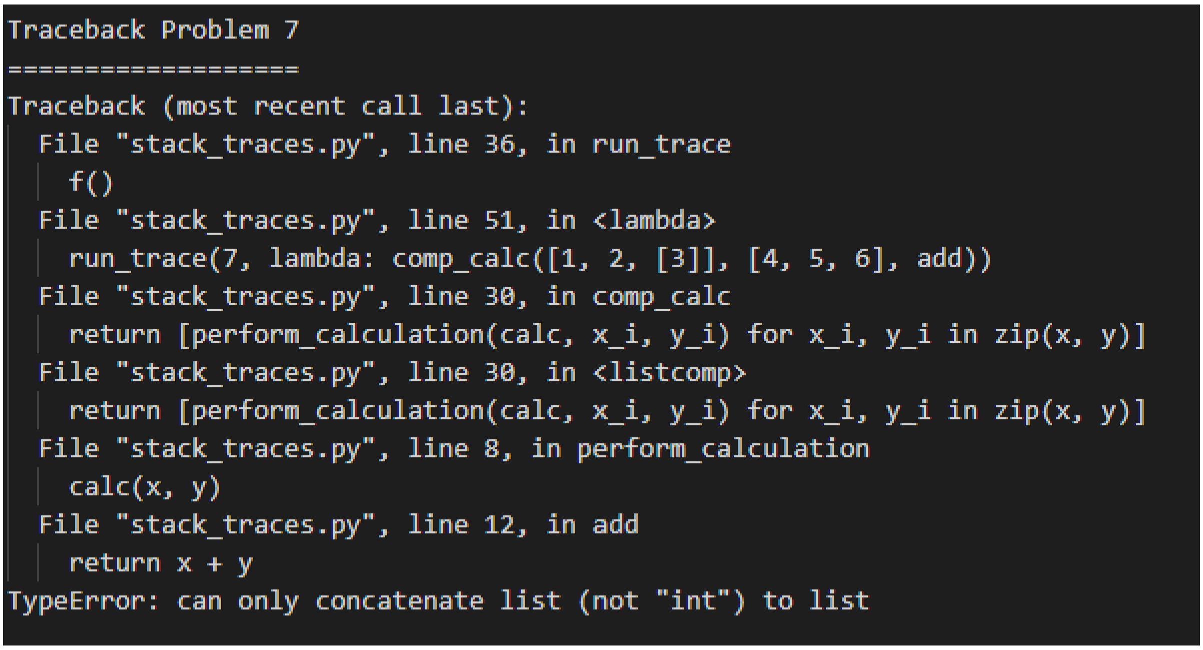
3.6 Trace 6



```
Traceback Problem 6
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 50, in <lambda>
    run_trace(6, lambda: comp_calc([1, 2, 3], 1, add))
  File "stack_traces.py", line 30, in comp_calc
    return [perform_calculation(calc, x_i, y_i) for x_i, y_i in zip(x, y)]
TypeError: zip argument #2 must support iteration
```

The issue here is that we are trying to iterate over something that can't be iterated on. I would look at line 30 to check types that are being passed in or transform what is being passed in to something iterable.

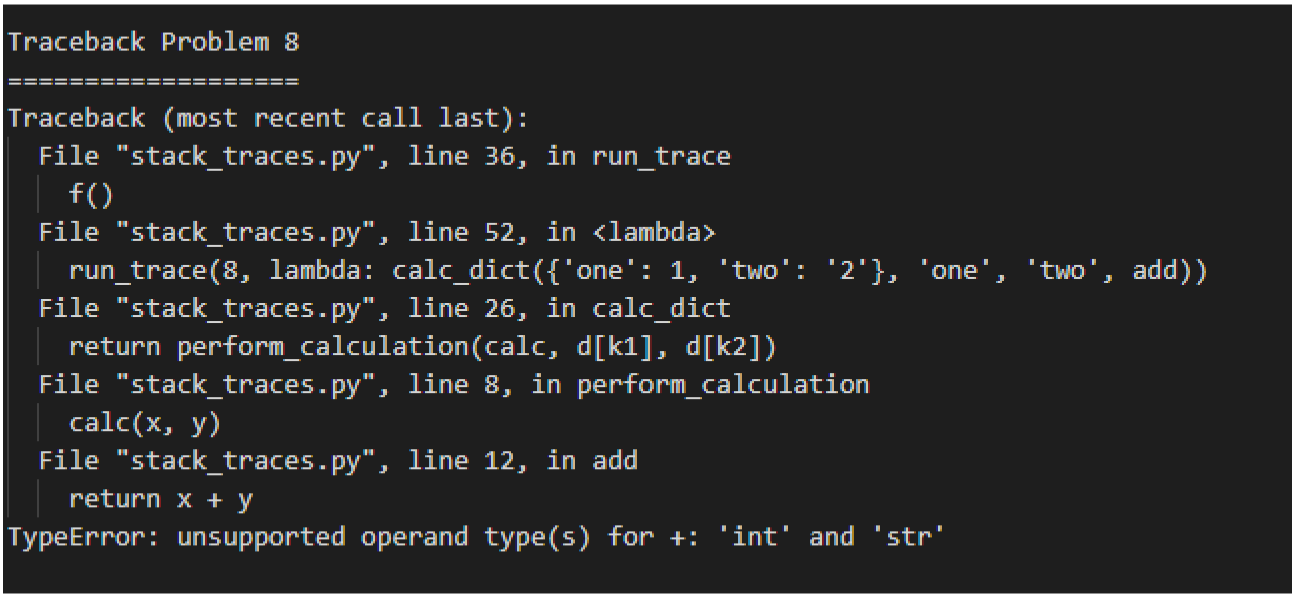
3.7 Trace 7



```
Traceback Problem 7
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 51, in <lambda>
    run_trace(7, lambda: comp_calc([1, 2, [3]], [4, 5, 6], add))
  File "stack_traces.py", line 30, in comp_calc
    return [perform_calculation(calc, x_i, y_i) for x_i, y_i in zip(x, y)]
  File "stack_traces.py", line 30, in <listcomp>
    return [perform_calculation(calc, x_i, y_i) for x_i, y_i in zip(x, y)]
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 12, in add
    return x + y
TypeError: can only concatenate list (not "int") to list
```

The issue here seems to be that we are trying to concat a list and a int. I would look at like 30 to check types that are being passed in.

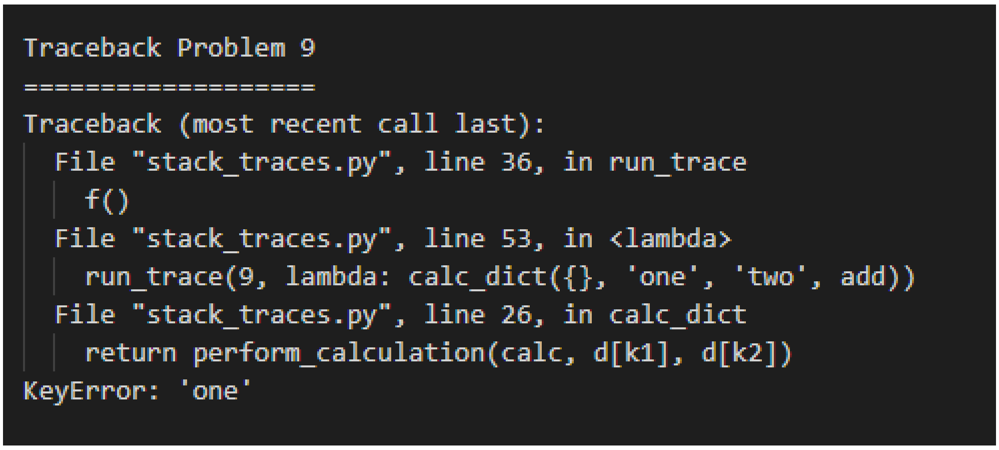
3.8 Trace 8



```
Traceback Problem 8
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 52, in <lambda>
    run_trace(8, lambda: calc_dict({'one': 1, 'two': '2'}, 'one', 'two', add))
  File "stack_traces.py", line 26, in calc_dict
    return perform_calculation(calc, d[k1], d[k2])
  File "stack_traces.py", line 8, in perform_calculation
    calc(x, y)
  File "stack_traces.py", line 12, in add
    return x + y
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

The issue here is that we are trying to conact a int and a string, I would look at line 26, it seems like we are assuming that all entries in the dict will be a int or a string and we are not handling the case when they are not.

3.9 Trace 9



```
Traceback Problem 9
===================
Traceback (most recent call last):
  File "stack_traces.py", line 36, in run_trace
    f()
  File "stack_traces.py", line 53, in <lambda>
    run_trace(9, lambda: calc_dict({}, 'one', 'two', add))
  File "stack_traces.py", line 26, in calc_dict
    return perform_calculation(calc, d[k1], d[k2])
KeyError: 'one'
```

The issue here is that we are trying to use the key ”one” when it does not seem to exist. To fix this we need to check that ”one” is present, I would look at line 26 to do this check, or line 53 if we wanted to modify the input to make more sense.