Assignment 2

Challenge: Python Stack Trace Interpretation

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.

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
```

Problem description:

Error:
`TypeError` - the function call or operands are of incompatible types
Error Message - `can only concatenate str (not "int") to str`

In python we need to pass the operands of the same type since Python is not doing any casting. In this case, since the first parameter is a string, it will try do a string concatenation, which fails because the second parameter is evaluated as int.

Solution:
Line 45: Fix the function call by either passing 2 strings or 2 number types

---

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'
```

Problem description:

Error:
TypeError - the function call or operands are of incompatible types
Error Message - `unsupported operand type(s) for +: 'int' and 'str'`

The same as the previous trace, but in this case is evaluating the first parameter as an integer, and math addition function will not work with the second parameter being a string..

Solution:
Line 46: Fix the function call by either passing 2 strings or 2 number types

---

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'
```

Problem description:

Error:
TypeError - the function call or operands are of incompatible types
Error Message - `can't multiply sequence by non-int of type 'str'`

The function will take 2 parameters and it will try to do the multiplication operation on those parameters. Because the passed parameters are string the multiplication will fail.

Solution:
Line 47: Fix the function call by passing 2 numeric values

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'

Problem description:

Error:
`TypeError` - the function call or operands are of incompatible types
Error Message - `can't multiply sequence by non-int of type 'list'`

Same as the issue above, but in this case the parameters are 2 arrays of ints. The * operator will not work on lists.

Solution:
Line 48:

- Fix the function call by passing 2 numeric values
- Fix parameter to pas the actual value in the array like [4][0] and [3][0]
- Fix parameters to have one array and one integer. ex: [4] and 3

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
```

Problem description:

Error:

`ValueError` - custom generated error

Error Message - `can't multiply sequence by non-int of type 'list'`

This is an error thrown from a try catch block.

Solution:

- Investigate the reason in function spelunk(), at line 21

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
```

Problem description:

Error:

`TypeError` - the function call or operands are of incompatible types

Error Message - `zip argument #2 must support iteration`

The zip function takes 2 arguments of type iterable and produces a list of tuples, each tuple containing the first element from the first iterable, paired with the first element from the second iterable,and so on.

Solution:
Line 50: Fix the function call by passing 2 iterable values

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
```

Problem description:

Error:
`TypeError` - the function call or operands are of incompatible types
Error Message - `can only concatenate list (not "int") to list`

The code tries to execute an addition between the tuples elements and fails when one of the tuple elements is a list, in this case [3].

Solution:
Line 51: Fix the function call by passing 2 arrays of int

```
Example: comp_calc([1, 2, 3], [4, 5, 6], add))
```

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'
```

Problem description:

Error:

`TypeError` - the function call or operands are of incompatible types

Error Message - `unsupported operand type(s) for +: 'int' and 'str'`

The function tries to add the values extracted from the dictionary object and do an addition operation on those. The problem is that the two extracted values are one string and one int . and the addition operation will not work

Solution:

Line 52: Fix the function call by passing either 2 strings or to integers in the dict object

`Example: {'one': '1', 'two': '2'} or {'one': 1, 'two': 2}`

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'
```

Problem description:

Error:
`KeyError` - key error
Error Message - `one`

The call fails because the dict object is empty

Solution:
Line 53: Fix the function call by passing non empty dict object