Q1:

Recommended Time 25 min

- Type something so that Python gives a NameError.
- Type something so that Python gives a SyntaxError.
- Type something so that Python gives a TypeError.
- Type something so that Python gives a IndexError.
- Type something so that Python gives a KeyError.
- Type something so that Python gives a AttributeError.
- Type something so that Python gives a ValueError.

Q2:

Recommended Time 10 min

Create a program that prompts the user for a number, and then uses a try-except block to catch any exceptions that might occur when converting the input to an integer. Use the finally keyword to display a message indicating whether the input was successfully converted or not.

Q3:

Recommended Time 10 min

Define a function capitalize_last_name() that accepts as argument a string with a (single) first and a (single) last name, and returns a string in which only the first letter of the first name is uppercase, whereas all letters of the last name are uppercase; in other words, 'marisa tomei' becomes 'Marisa TOMEI'. (Tip: use str.split() to split a str into separate words.)

If something other than a str object is passed as an argument, the function should raise a TypeError. (Tip: you can use isistance() to check whether an object is of a particular type.) If the str does not consist of exactly two words, the function should raise a ValueError.

Q4:

Recommended Time 15 min

You're going to write an interactive calculator! User input is assumed to be a formula that consist of a number, an operator (at least + and -), and another number, separated by white space (e.g. 1 + 1). Split user input using str.split(), and check whether the resulting list is valid:

- If the input does not consist of 3 elements, raise a FormulaError, which is a custom Exception.
- Try to convert the first and third input to a float (like so: float_value = float(str_value)). Catch any ValueError that occurs, and instead raise a FormulaError
- If the second input is not '+' or '-', again raise a FormulaError

If the input is valid, perform the calculation and print out the result. The user is then prompted to provide new input, and so on, until the user enters quit.

An interaction could look like this:

```
>>> 1 + 1
2.0
>>> 3.2 - 1.5
1.7000000000000002
>>> quit
```

Q5:

Recommended Time 10 min

Create a function called safe_int() that takes a single argument i. If possible, the function converts i to int and returns it. If not possible (i.e. if an Exception occurs), the function returns None.

Q6:

Recommended Time 10 min

Place msg="You can't add int to string" to the right place so that program avoids BaseExceptionError.

You can use except Exception although normally you should be careful using such powerful exception statements.

```
a="Hello World!"
try:
    a + 10

print(msg)
```

Q7:

Recommended Time 10 min

Write a program that takes two inputs from the user: a list of numbers and a number. The program should assert that the number entered by the user is in the list of numbers and its index is even. If the assertion is true, print "Number found in list with even index". If the assertion is false, print "Number found in list with odd index or not found in list".