Exercise 1

**ESTIMATED TIME TO COMPLETE: 6 minutes**

Try to answer the following questions by just reading the code. Reading code is a very good skill to have (and will help you both in your programming career and on your exams!). It is okay to check your answers that you obtain from just reading the code, then in your interpreter run the code for the ones you got wrong on your first attempt.

What error (if any) is raised when the following code snippets are attempted?

* '1' / '2'

SyntaxError

ValueError

TypeError

NameError

No error is raised

* '1' / 2

SyntaxError

ValueError

TypeError

NameError

No error is raised

* int('1') / 2.0

SyntaxError

ValueError

TypeError

NameError

No error is raised

* mylist = [10, 20, 30]  
  mylist.index(11)

SyntaxError

ValueError

TypeError

NameError

No error is raised

* A=2  
  3\*a

SyntaxError

ValueError

TypeError

NameError

No error is raised

Exercise 2

**ESTIMATED TIME TO COMPLETE: 14 minutes**

Below are some short Python programs. For each program, answer the associated question.

Try to answer the questions without running the code. Check your answers, then run the code for the ones you get wrong.

These questions will ask you to write what the code prints out. If an exception is raised that is not handled by the code write "error" (no quotes), in addition to any other text that is output.

The function in the following questions takes a list of integers numbersand a position index, and divides each entry in the list of numbers by the value at entry index.

Write what it prints out, separating what appears on a new line by a comma and a space.

def fancy\_divide(numbers,index):

try:

denom = numbers[index]

for i in range(len(numbers)):

numbers[i] /= denom

except IndexError:

print("-1")

else:

print("1")

finally:

print("0")

What does fancy\_divide([0, 2, 4], 1) print out?



What does fancy\_divide([0, 2, 4], 4) print out?



What does fancy\_divide([0, 2, 4], 0) print out?



def fancy\_divide(numbers, index):

try:

denom = numbers[index]

for i in range(len(numbers)):

numbers[i] /= denom

except IndexError:

fancy\_divide(numbers, len(numbers) - 1)

except ZeroDivisionError:

print("-2")

else:

print("1")

finally:

print("0")

What does fancy\_divide([0, 2, 4], 1) print out?



What does fancy\_divide([0, 2, 4], 4) print out?



What does fancy\_divide([0, 2, 4], 0) print out?



def fancy\_divide(numbers, index):

try:

try:

denom = numbers[index]

for i in range(len(numbers)):

numbers[i] /= denom

except IndexError:

fancy\_divide(numbers, len(numbers) - 1)

else:

print("1")

finally:

print("0")

except ZeroDivisionError:

print("-2")

What does fancy\_divide([0, 2, 4], 1) print out?



What does fancy\_divide([0, 2, 4], 4) print out?



What does fancy\_divide([0, 2, 4], 0) print out?



def fancy\_divide(list\_of\_numbers, index):

try:

try:

raise Exception("0")

finally:

denom = list\_of\_numbers[index]

for i in range(len(list\_of\_numbers)):

list\_of\_numbers[i] /= denom

except Exception as ex:

print(ex)

Does this code print 0 when you call fancy\_divide([0, 2, 4], 0)?

Yes.

No.

def fancy\_divide(list\_of\_numbers, index):

try:

try:

denom = list\_of\_numbers[index]

for i in range(len(list\_of\_numbers)):

list\_of\_numbers[i] /= denom

finally:

raise Exception("0")

except Exception as ex:

print(ex)

Does this print 0 when you call fancy\_divide([0, 2, 4], 0)?

Yes.

No.

Exercise: simple divide

5/5 points (graded)

**ESTIMATED TIME TO COMPLETE: 4 minutes**

Suppose we rewrite the FancyDivide function to use a helper function.

def fancy\_divide(list\_of\_numbers, index):

denom = list\_of\_numbers[index]

return [simple\_divide(item, denom) for item in list\_of\_numbers]

def simple\_divide(item, denom):

return item / denom

This code raises a ZeroDivisionError exception for the following call: fancy\_divide([0, 2, 4], 0)

Your task is to change the definition of simple\_divide so that the call does not raise an exception. When dividing by 0, fancy\_divide should return a list with all 0 elements. Any other error cases should still raise exceptions. You should only handle the ZeroDivisionError.

# still takes same arguments

def simple\_divide(item, denom):

# start a try-except block

try:

return item / denom

# catch a division by zero and return 0

except ZeroDivisionError:

return 0

### Exercise 3

**ESTIMATED TIME TO COMPLETE: 6 minutes**

Consider the function normalize that takes as input a list of positive numbers numbers and returns a list of numbers that are a fraction of the maximum element in the list. Try to answer the questions without running the code. Check your answers, then run the code for the ones you get wrong. You'll learn the most this way, by figuring things out, instead of just running the code and reading off the answers.

def normalize(numbers):

max\_number = max(numbers)

for i in range(len(numbers)):

numbers[i] /= float(max\_number)

return numbers

The code below tries to call normalize with one particular input. Answer the next 5 questions based on the following code.

try:

normalize([0, 0, 0])

except ZeroDivisionError:

print('Invalid maximum element')

1. Does the try block throw (also known as raise) an exception?

Yes

No

correct

1. What is the name of the exception the code is trying to catch?



1. What is the output?



1. Since we are dividing by the maximum element in a list of positive numbers, we know that normalize will return a value between 0 and 1. What type of condition is this?

pre condition

post condition

1. We also know the result is not meaningful when the maximum element is 0, so we want to ensure that the numbers in the list do not violate this. What type of condition is this?

pre condition

post condition

Now assume the definition of the function normalize is rewritten as follows

def normalize(numbers):

max\_number = max(numbers)

assert(max\_number != 0), "Cannot divide by 0"

for i in range(len(numbers)):

numbers[i] /= float(max\_number)

assert(0.0 <= numbers[i] <= 1.0), "output not between 0 and 1"

return numbers

Answer the next 3 questions based on this code.

1. Which condition does the line assert(max\_number != 0) correspond to?

pre condition

post condition

1. Which condition does the line assert(0.0 <= numbers[i] <= 1.0) correspond to?

pre condition

post condition

1. What does the function call normalize([0, 0, 0]) print out?

