CPS 109 - Lab 9

The Final Countdown

1. Welcome to the final lab for CPS109!
2. No lab next week 😢
3. Will have an **optional review session** on Tuesday November 29th from 8-10 am in ENG 201, and 10am-12pm in ENG 201/202 (will do some exam review, AMA, etc.)
4. Note that this is unofficial and unsanctioned

Agenda

1. A2 Testing Details
2. ~~Dictionaries~~ (already done!)
3. Randomness
4. Multi-Dimensional Arrays
5. Exception Handling

A2 Testing

What to do:

Download the problem descriptions and the automated tester from Ilkka's GitHub here:

https://github.com/ikokkari/PythonProblems

Click the green Code button, then download as zip. Extract the contents of this archive somewhere on your computer. Open labs109.py and tester109.py in your IDE of choice. You'll notice that labs109.py already has the solution to the Ryerson letter grade problem in it, which is why this one is not eligible.

When you want to test your functions with the tester, you will run the tester109.py file. If you run it immediately, you will notice that the provided solution for Ryerson letter grade passes. Not bad!

If you got this far, you're ready to start choosing problems from the PDF file that comes with the zip archive. Solve these problems in the same labs109.py file. As you add solutions, the tester will report which have passed and which have failed.

Sample A2 Output

```
109 Python Problems tester, November 3, 2022, Ilkka Kokkarinen.
Student file labs109.py contains 3 recognized functions to test.
Finished reading expected answers.
ryerson_letter_grade: Success in 0.001 seconds.
is_ascending: Success in 0.155 seconds.
only_odd_digits: DISCREPANCY AT TEST CASE #1:
ARGUMENTS: 12
EXPECTED: False
RETURNED: True
2 out of 3 functions of 123 possible work.
[Finished in 248ms]
```

A2 Cont.

- Due December 5th
- Use Python 3.7+ (mandatory)
- 10 questions
- Pass/fail for each question (no part marks - no comments needed)
- Submit labs109.py (named that way!)

Randomness

What is randomness? Well, in python, it’s using pseudo randomly generated values or choices to use in your program.

Randomness

Why would you use random numbers or choices?

Several reasons: games of cards, rolling dice, etc.

Multi-Dimensional Arrays

You all have dealt with matrices by now, right?

Multi-dimensional arrays are a way to represent those.

Multi-Dimensional Arrays

And what are those?

They’re just lists within lists.

Exception Handling

- It is often the case that your program will run into an error, especially when dealing with user inputs

When to Use It

- Whenever you want to ensure that your program can continue running even when an error has occurred, or you want the program to halt upon encountering a certain situation
- E.g. to keep running until the user enters some valid inputs

Try/Except (see CrashCourse)

```python
# Typically dividing by 0 gives an error. Let's use a try/except block to catch this error.
try: # We will try to run whatever is put in here.
    print("Trying bad division") # Notice that this prints
    x = 1/0 # Illegal operation
    print(x) # Notice that this doesn't print
except: # When we get an exception of any kind we run this block
    print("An exception has occurred") # We can print, or do whatever here. Typically it is good practice to print/return something.
```

```python
# If we want to be more detailed re: what kind of error we're getting, we can use the following syntax
try:
    print("Trying bad division again")
    y = 1/0
    print(y)
except Exception as err: # Exception is the wildcard class to catch (most) non-fatal exception
    print(f"Unexpected {err=}, {type(err)=}")
    #raise # We can optionally call raise here, to halt the execution of our program upon encountering an error
```

Try/Except/Except

```
# This third way of doing things is the best practice -
# to anticipate certain kinds of errors and proceed depending on this -
# and raise an exception if we encounter something unexpected
try:
    print("Trying bad division finally")
    z = zz/1 # Uncomment to get name error
    #z = 1/0 # Uncomment to get zero division error
    #q = []
    #z = q[1] # Uncomment to get unhandled error
    print(z)
except NameError as err:
    print("Name Error:", err)
    z = 1
except ZeroDivisionError as err:
    print("Zero Divison Error:", err)
    z = 2
except Exception as err:
    print(f"Unexpected {err=}, {type(err)=}")
    raise # Raise the error and halt execution
print(z) # Notice how this is different depending on which except was hit
```

Try/Except/Else

```
# Let's now see a common use-case: error handling for user input!
valid_input_entered = False
while not valid_input_entered: # While the user has not entered valid inputs for division...
    try:
        numerator = int(input("Enter your numerator (int):")) # Try to cast their numerator to an int
        denominator = int(input("Enter your denominator (int):")) # Try to cast their denominator to an int
        division_result = numerator/denominator # Try to divide numerator by denominator
        #division_result = aaa/denominator # Uncomment to see the base exception case
        print("Good inputs!")
    except ValueError as err:
        print("Please enter two integers:", err)
    except ZeroDivisionError as err:
        print("Please ensure your denominator is not 0:", err)
    except Exception as err:
        print(f"Unexpected {err=}, {type(err)=}")
        raise # Break out
    else: # Your else will occur after your try if no exceptions have occurred
        print("The division result is:", division_result)
        valid_input_entered = True
```

Try/Except/Finally

```
# Last is using the finally case
a = 1
#b = 0
b = 1 # Uncomment to see how this works with no exception raised

try:
    c = a/b
    print(c)
except ZeroDivisionError as err:
    print("Zero Divison Error:", err)
finally: # Stuff in the finally will always occur right before the try statement completes (and right before the except completes if there is an exception)
    print("Finally is happening now")
    c = 0 # Mwahaha

print(c)
```

Raise

```
# First is raising a specific exception
anything_but_five = float(input("Enter any number except 5:"))
if (anything_but_five == 5.0):
    raise ValueError("I said not to enter 5! >:(")
else:
    print("Well done friend!")
```

Thanks / Final Thoughts

- Thanks for coming to labs throughout the semester
- Best of luck with your exams (and make sure to study hard for CPS109)!
- Try to stay in touch with your peers
- Feel free to reach out anytime / add us on LinkedIn!