**Lab 6 - Files and Exceptions**

**Questions:**

**1. Given:****number\_str = input("Input a floating-point number: ")  
while True:  
 *# Line 1*print("Number is", number\_float)**

**Write a try/except block in # Line 1 that will keep prompting until a correctly formatted floating-point is entered.**

number\_str = input("Input a floating-point number: ")

while True:

try:

number\_float = float(number\_str)

break

except ValueError:

number\_str = input(“Error, enter a float: ”)

print("Number is", number\_float)

**2. Write a function named safe\_input(prompt, type) that works like the Python input function, except that it only accepts the specified type of input. The function takes two arguments:**

* **prompt: str**
* **type: int, float, str**

**The function will keep prompting for input until the correct input of the specified type is entered. The function returns the input. If the input was specified to be a number (float or int), the value returned will be of the correct type; that is, the function will perform the conversion. The default for a prompt is the empty string. The default for the type is string.**

def safe\_input(prompt, type):

while True:

try:

i = Type(input(prompt))

return

except ValueError:

print("invalid type and prompt")

prompt = input("Enter a new prompt")

print(safe\_input("Enter value", int))

c = safe\_input("a",int)

print("Return here", c)

**3. Write a function named prompt\_open that prompts for a file name and repeatedly attempts to read the specified file until a correctly specified file has been entered. The function takes one mode argument, 'r' or 'w', and returns the file handle that open returns.**

def prompt\_open(mode\_of\_operation):

while True:

file\_Name = input("Enter the file name to open:")

try:

fHandle = open(file\_Name, mode\_of\_operation)

return fHandle

except IOError:

print("File name does not exist on the specified path. Try again.")

**4. Write a program that prompts for three numbers. Divide the first number by the second number and add that result to the third number. Using exceptions check for the following errors: ValueError, and ZeroDivisionError.**

Three\_integers = input("Enter the three integers seperated by a space:")

x,y,z=Three\_integers.split()

try:

x=int(x)

y=int(y)

z=int(z)

d=x/y+z

except ValueError:

print("Occurrence of value Error:",x,y,z)

except ZeroDivisionError:

print("Occurrence of division by zero Error:", x,"/",y)

print(d)

**5. Given:**

***# reverse each line of the input file in the output file***

**file\_str = input("Open what file:")**

**find\_line\_str = input("Which line (integer):")**

**try:**

**input\_file = open(file\_str) *# potential user error***

**find\_line\_int = int(find\_line\_str) *# potential user error***

**line\_count\_int = 1**

**for line\_str in input\_file:**

**if line\_count\_int == find\_line\_int:**

**print("Line {} of file {} is {}".format(find\_line\_int, file\_str, line\_str))**

**break**

**line\_count\_int += 1**

**else:**

***# get here if line sought doesn't exist***

**print("Line {} of file {} not found".format(find\_line\_int, file\_str))**

**input\_file.close()**

**except IOError:**

**print("The file", file\_str, "doesn't exist.")**

**except ValueError:**

**print("Line", find\_line\_str, "isn't a legal line number.")**

**print("End of the program")**

**(a) When IOError occurred the user had to enter a line number before the error occurred. Rewrite the code so that if a bad file name is entered, the error will be handled before a line number is requested.**

**(b) Rewrite the code so that if IOError occurs the program keeps asking for input until the user gets it right.**

**(c) Rewrite the code so that if error ValueError occurs the program keeps asking for input until the user gets it right.**

(a) When IOError occurred the user had to enter a line number before the error occurred. Rewrite the code so that if a bad file name is entered, the error will be handled before a line number is requested.

file\_str = input("Open what file:")

try:

input\_file = open(file\_str) # potential user error

except IOError:

print("The file", file\_str, "doesn't exist.")

else:

find\_line\_str = input("Which line (integer):")

try:

find\_line\_int = int(find\_line\_str) # potential user error

line\_count\_int = 1

for line\_str in input\_file:

if line\_count\_int == find\_line\_int:

print("Line {} of file {} is {}".format(find\_line\_int, file\_str, line\_str))

break

line\_count\_int += 1

else:

# get here if line sought doesn't exist

print("Line {} of file {} not found".format(find\_line\_int, file\_str))

input\_file.close()

except ValueError:

print("Line", find\_line\_str, "isn't a legal line number.")

print("End of the program")

(b) Rewrite the code so that if IOError occurs the program keeps asking for input until the user gets it right.

while True:

file\_str = input("Open what file:")

try:

input\_file = open(file\_str) # potential user error

except IOError:

print("The file", file\_str, "doesn't exist.")

else:

break

while True:

find\_line\_str = input("Which line (integer):")

try:

find\_line\_int = int(find\_line\_str) # potential user error

break

except ValueError:

print("Line", find\_line\_str, "isn't a legal line number.")

line\_count\_int = 1

for line\_str in input\_file:

if line\_count\_int == find\_line\_int:

print("Line {} of file {} is {}".format(find\_line\_int, file\_str, line\_str))

break

line\_count\_int += 1

else:

# get here if line sought doesn't exist

print("Line {} of file {} not found".format(find\_line\_int, file\_str))

input\_file.close()

print("End of the program")

(c) Rewrite the code so that if error ValueError occurs the program keeps asking for input until the user gets it right.

file\_str = input("Open what file:")

while True:

try:

input\_file = open(file\_str) # potential user error

break

except IOError:

print("The file", file\_str, "doesn't exist.")

file\_str = input("Open what file:")

while True:

find\_line\_str = input("Which line (integer):")

try:

find\_line\_int = int(find\_line\_str) # potential user error

break

except ValueError:

print("Line", find\_line\_str, "isn't a legal line number.")

line\_count\_int = 1

for line\_str in input\_file:

if line\_count\_int == find\_line\_int:

print("Line {} of file {} is {}".format(find\_line\_int, file\_str, line\_str))

break

line\_count\_int += 1

else:

# get here if line sought doesn't exist

print("Line {} of file {} not found".format(find\_line\_int, file\_str))

input\_file.close()

print("End of the program")