**KENDRIYA VIDHYALAYA**

**CHANDKHEDA, AHMEDABAD**



**PROJECT REPORT ON**

**ORGANIC CHEMISTRY DATABASE**

**SUBMITTED TO SUBMITTED BY:**

**MR. JITENDRA VARATIYA HARSHIL SOLANKI**

**P.G.T.(COMP. SC) & VANSH KANODIYA**

|  |
| --- |

INDEX

1. CERTIFICATE
2. ACKNOWLEDGEMENT
3. INTRODUCTION
4. SOFTWARE/HARDWARE

REQUIREMENT

5. DATABASE DESCRIPTION

6. SOURCE CODE

7. OUTPUT

8. BIBLIOGRAPHY



**CERTIFICATE**

THIS IS TO CERTIFY THAT HARSHIL SOLANKI OF CLASS XII, ROLL NO 12117 HAS SUCCESSFULLY COMPLETED PROJECT TITLED ORGANIC CHEMISTRY DATABASE FOR SUBJECT OF COMPUTER SCIENCE NEW (083) FOR FULFILLMENT OF AISSCE PRACTICAL EXAMINATION 2022-23.

INTERNAL EXAMINER EXTERNAL EXAMINER

PRINCIPAL

**ACKNOWLEDGMENT**

It is with pleasure that I acknowledge my sincere gratitude to our teacher, MR. JITENDRA VARATIYA SIR who taught and undertook the responsibility of teaching the subject computer science. I have greatly benefited from his classes.

I am especially indebted to our Principal MR. ASHOK RATHI SIR who has always been a source of encouragement and support and without whose inspiration this project would not have been a successful I would like to place on record heartfelt thanks to him.

Finally, I would like to express my sincere appreciation for all the other students of my batch, their friendship & the fine times that we all shared together.

**INTRODUCTION**

This project aims to provide a user-friendly python interface to deal with the chemical reactions, their reactants, reagents, products etc. of the organic chemistry branch of chemistry. Using this code, users will be able to search the database using any term of the reaction, which may prove helpful in solving questions, users would no longer have to search books or notebooks to find out the desired chemical reaction. The user will be able to expand the database by inserting new reactions into the database, and could save the changes in database into a sql file for extended use. The user will be able to insert brief remarks over the reaction, its mechanism and any exceptions for the reagents or reactants as per case.

The project is written completely in python programming language and it uses python interface at the front-end and the mysql database at the back-end. Function definitions are used to reduce the length of the code and provide a convenient way for the program to be inspected. It includes use of some mysql commands to fetch data or execute operations in the back-end, i.e. MySql. It is easy to use.

**HARDWARE AND SOFTWARE REQUIREMENTS**

* **Modern Operating System:** Windows 7 to 11
* **X86 64-bit CPU:** Intel/AMD Architecture
* **4 GB** minimum RAM Support
* **5 GB** minimum Free Disk Space (For Smooth Running of Program)
* MySql should be supported on the operating system and Database should be accessible by the user.

**DATABASE DESCRIPTION**

The database is named Organic\_chem containing the table Reactions containing Columns Name (with constraint NOT NULL), Reactant, Reagent (NOT NULL), Conditions, Product, Remarks. Column Remarks has datatype TEXT while all other columns have datatype TINYTEXT. TEXT is a data type with a field with a maximum length of 65535 characters. Sorts and comparisons on stored data are not case sensitive in TEXT fields. We do not need to specify a length with TEXT. TINYTEXT is Text datatype. It is a TEXT column with a maximum length of 255 characters. We do not need to specify a length with TINYTEXT.

**CODE**

import mysql.connector

import os

dir\_path = os.path.dirname(os.path.realpath(\_\_file\_\_))

cnx = mysql.connector.connect(host='localhost', port='2836', user='root', passwd='root', database = 'Organic\_chem')

cur = cnx.cursor()

\_format\_ = ['Name', 'Reactant', 'Reagent', 'Conditions', 'Product', 'Remarks']

class helpers():

def \_print\_rx\_(rx):

print(\*('-->'+key+':- '+item for item, key in zip(rx,\_format\_)), sep='\n')

def search\_term(term):

'''OLD CODE DISCARDED

term1 = input("Enter %s of the reaction you want to search:- "%term)

if term1 in rx[n]:

print("The Reaction is as follows:")

helpers.\_print\_rx\_(rx)

return rx

print("The Reaction having such %s could not be found"%term)

return None

'''

term1 = input("Enter %s of the reaction you want to search:- "%term)

cur.execute("select \* from Reactions where {} LIKE '%{}%';".format(term, term1))

rx = cur.fetchall()

if rx == []:

print("The Reaction having such %s could not be found"%term)

helpers.\_continue\_()

return None

elif len(rx)== 1:

helpers.\_print\_rx\_(rx[0])

helpers.\_continue\_()

return rx[0]

else:

n = len(rx)

print('There are the following {} results: '.format(n))

helpers.\_continue\_()

for i in range(0, n):

helpers.page\_break()

print('<REACTION {}>'.format(i+1))

helpers.\_print\_rx\_(rx[i])

helpers.\_continue\_()

return rx

def change\_term(term, name):

term2 = input("Enter new %s for the Reaction:- "%term)

cur.execute("update Reactions set %s='%s' where 'Name'='%s';"%(term, term2, name))

cnx.commit()

print("Reaction successfully updated.")

return True

def page\_break():

print('-'\*120)

def \_continue\_():

input("Press ENTER to Continue")

class Ops():

def insert\_rx():

print('<INSERTING A REACTION>')

v1 = str(input("Enter name:- "))

v2 = str(input("Enter reactant(s):- "))

v3 = str(input("Enter reagent(s):- "))

v4 = str(input("Enter condition(s):- "))

v5 = str(input("Enter product(s):- "))

v6 = str(input("Remarks over the reaction (mechanism, explanation, reactivity,...):- "))

st = "insert into Reactions values('{}','{}','{}','{}','{}','{}');".format(v1,v2,v3,v4,v5,v6)

cur.execute(st)

cnx.commit()

print("Reaction successfully inserted into the database.")

return True

def list\_db():

print('<LISTING THE DATABASE>')

cur.execute('select \* from Reactions;')

data = cur.fetchall()

i = 1

for rx in data:

print('<Reaction %d>'%i)

helpers.\_print\_rx\_(rx)

helpers.page\_break()

i+=1

if data == []:

print("Database is Empty.")

helpers.\_continue\_()

return

def search\_db():

print('<SEARCHING THE DATABASE>')

print('''By which parameter would you like to search for reaction?

1. Name 2. Reactant 3. Reagent 4. Conditions

5. Product (6. Remarks (searches same as 7.)) 7. Search in all

''')

'''OLD CODE DISCARDED

cur.execute('select \* from Reactions;')

data = cur.fetchall()'''

i = input("Enter your choice (1-6):- ")

helpers.page\_break()

#for rx in data:

if int(i) == 1:

return helpers.search\_term('Name')

if int(i) == 2:

return helpers.search\_term('Reactant')

if int(i) == 3:

return helpers.search\_term('Reagent')

if int(i) == 4:

return helpers.search\_term('Conditions')

if int(i) == 5:

return helpers.search\_term('Product')

if int(i) == 6 or 7:

a = input("Enter the term present in the reaction:- ")

cur.execute('select \* from Reactions;')

data = cur.fetchall()

for rx in data:

for el in rx:

if a in el:

print("The Reaction is as follows:")

helpers.\_print\_rx\_(rx)

return rx

print("None of the Reactions contain this term")

return None

else:

print("There was an error in the input, execute again")

return None

def update\_db():

print('<UPDATING A REACTION')

print("First search for the reaction you want to update: ")

def sub\_update(rx):

'''COMMON PROCESS TO BE USED'''

# rx = list(rx) ---Used in old code, no longer needed.

i = int(input("Enter the term you want to update(1-6):- "))

if i == 1:

helpers.change\_term('Name', rx[0])

if i == 2:

helpers.change\_term('Reactant', rx[0])

if i == 3:

helpers.change\_term('Reagent', rx[0])

if i == 4:

helpers.change\_term('Conditions', rx[0])

if i == 5:

helpers.change\_term('Product', rx[0])

if i == 6:

helpers.change\_term('Remarks', rx[0])

helpers.\_continue\_()

return

try:

rx = Ops.search\_db()

if type(rx[0])== tuple:

n = int(input('Which one of the reactions would you like to update?(1-{}):- '.format(len(rx))))

return sub\_update(rx[n-1])

else:

return sub\_update(rx)

except ValueError or TypeError:

return

def delete\_rx():

print('<DELETING A REACTION>')

print("First search for the reaction you want to delete: ")

def sub\_del(rx):

cur.execute("delete from Reactions where {}='{}';".format('Name', rx[0]))

cnx.commit()

helpers.page\_break()

print("Reaction successfully deleted.")

helpers.\_continue\_()

return

try:

rx = Ops.search\_db()

if type(rx[0])==tuple:

n = int(input('Which one of the reactions would you like to delete?(1-{}):- '.format(len(rx))))

return sub\_del(rx[n-1])

else:

return sub\_del(rx)

except ValueError or TypeError:

return

def Naming():

f = open('{}\\datafiles\\IUPAC\_A.txt'.format(dir\_path), 'r')

data = f.read()

print(data)

f.close()

helpers.\_continue\_()

helpers.page\_break()

f = open('{}\\datafiles\\IUPAC\_B.txt'.format(dir\_path), 'r')

data = f.read()

print(data)

return

def OR():

f = open('{}\\datafiles\\OR\_C.txt'.format(dir\_path), 'r')

data = f.read()

print(data)

f.close()

helpers.\_continue\_()

helpers.page\_break()

f = open('{}\\datafiles\\OR\_D.txt'.format(dir\_path), 'r')

data = f.read()

print(data)

return

class Interface():

def operations():

helpers.page\_break()

print('''

Operations:-

1. Add Reaction to database 2. List out the database 3. Search the database

4. Update a reaction 5. Delete a reaction 6. IUPAC Nomenclature

7. Oxidations & Reductions 8. Exit

Select the operation you want to perform (1-8): ''', end='')

try:

i = int(input())

helpers.page\_break()

if i == 1:

return Ops.insert\_rx()

elif i == 2:

return Ops.list\_db()

elif i == 3:

return Ops.search\_db()

elif i == 4:

return Ops.update\_db()

elif i == 5:

return Ops.delete\_rx()

elif i == 6:

return Ops.Naming()

elif i == 7:

return Ops.OR()

elif i == 8:

global consent

consent = False

return

except TypeError:

print("There was error in the input. Execute again.")

return

def \_init\_():

print('''

----------------------------------------------------------------

###############################################################

|>---<|This is a database of Organic Chemistry Reactions|>---<|

###############################################################

----------------------------------------------------------------

''')

global consent

consent = True

while consent == True:

Interface.operations()

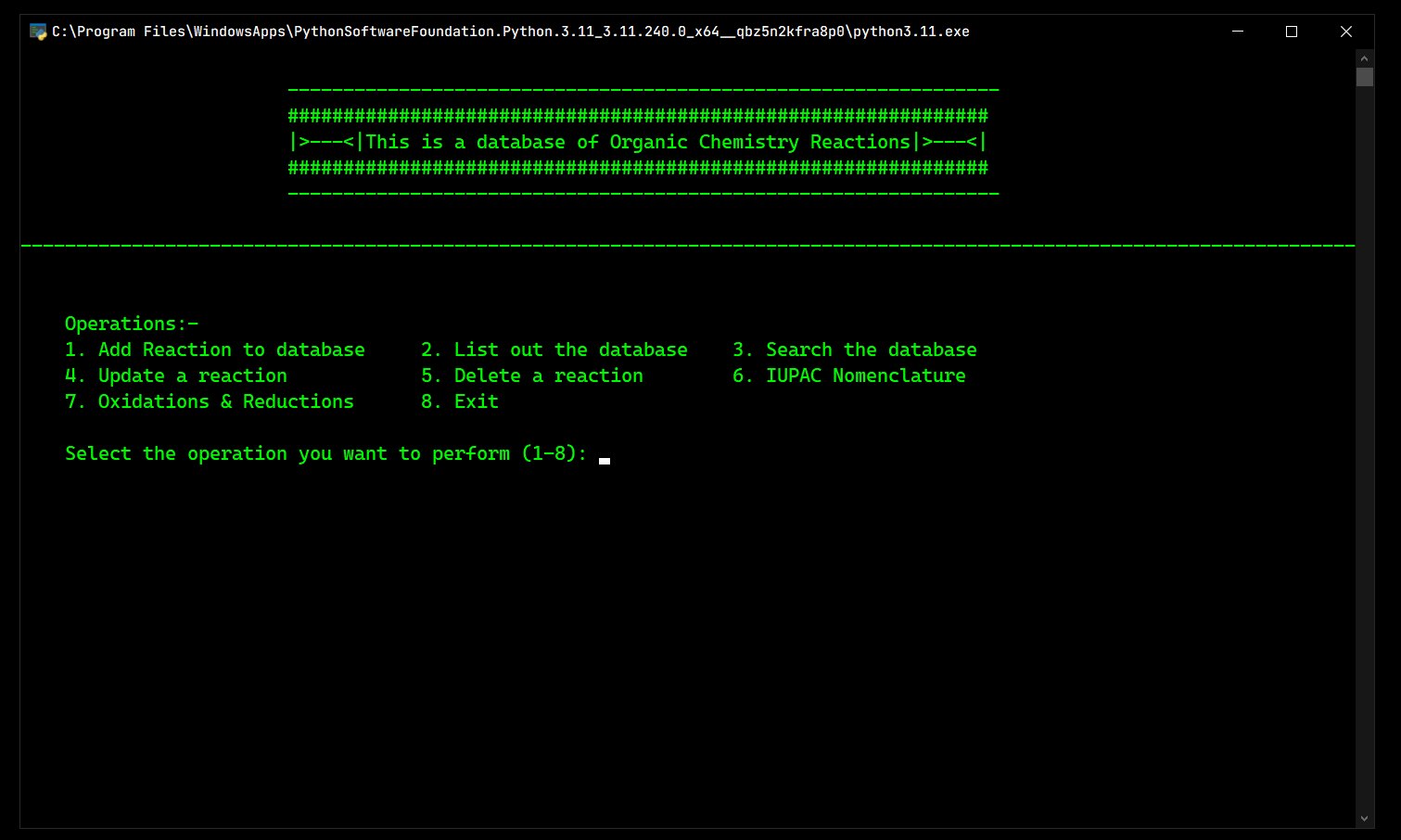
#INITIATION OF CODE

Interface.\_init\_()

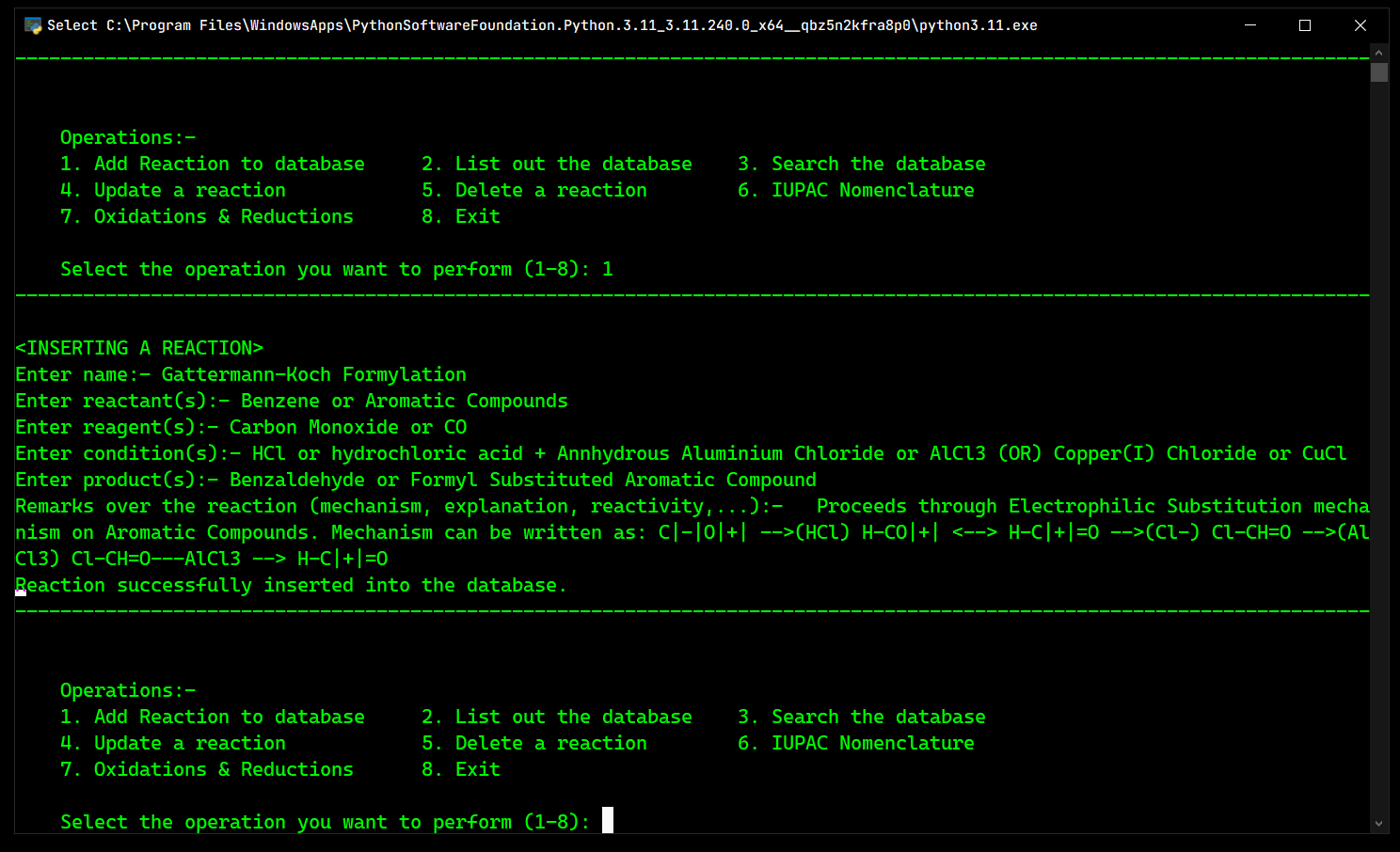
**—--------------------------------------------------CODE END—------------------------------------------------**

**OUTPUT**

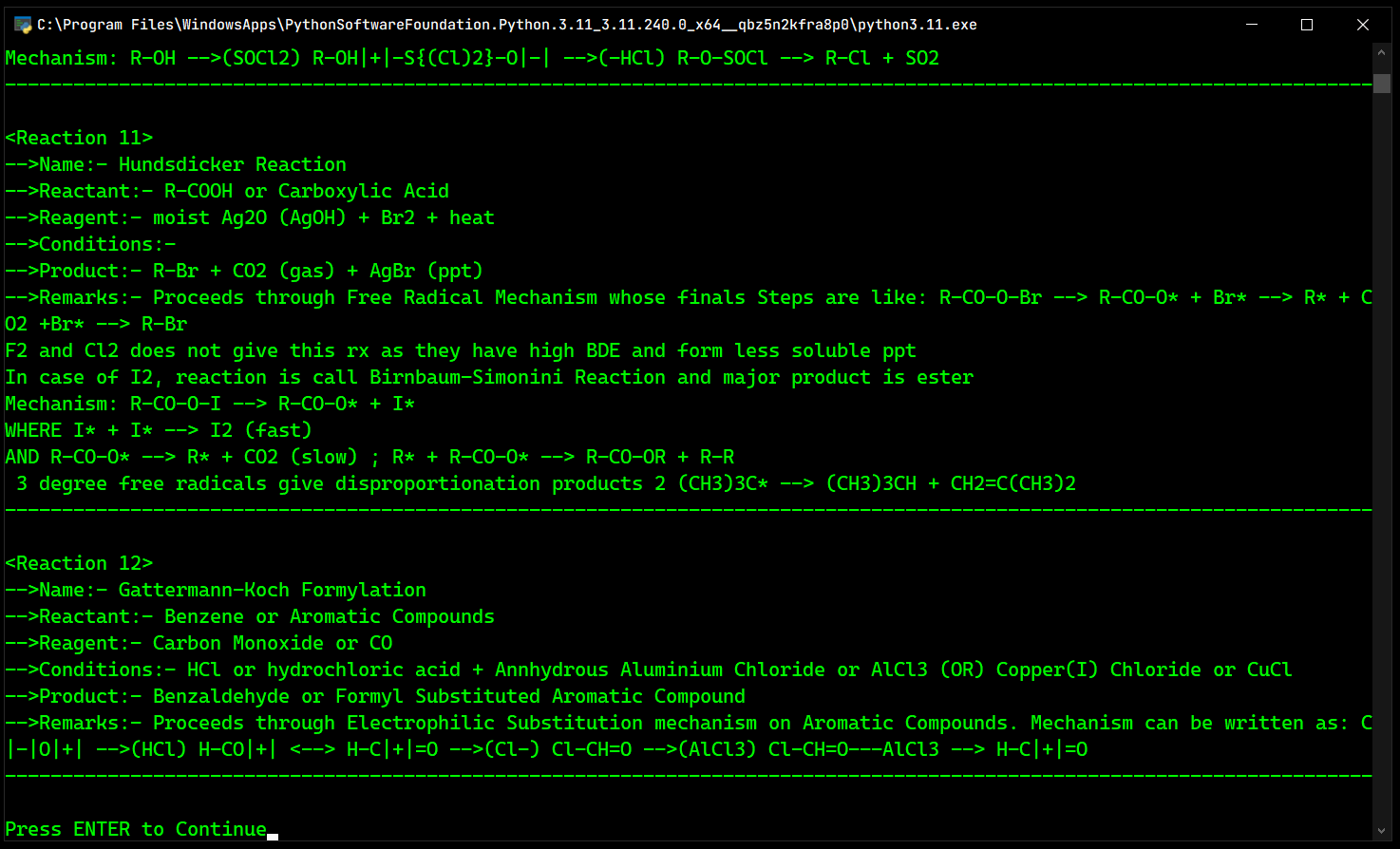
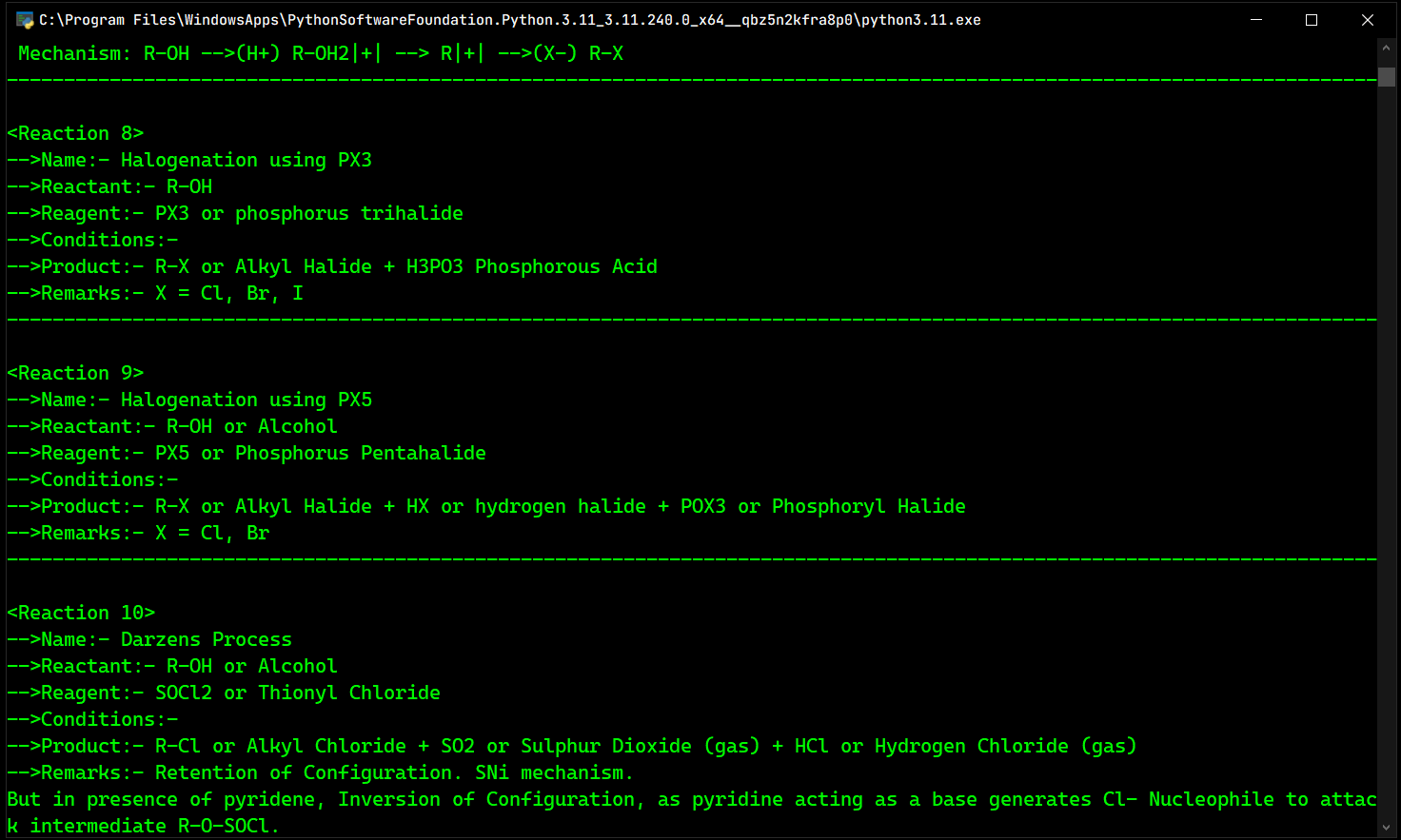
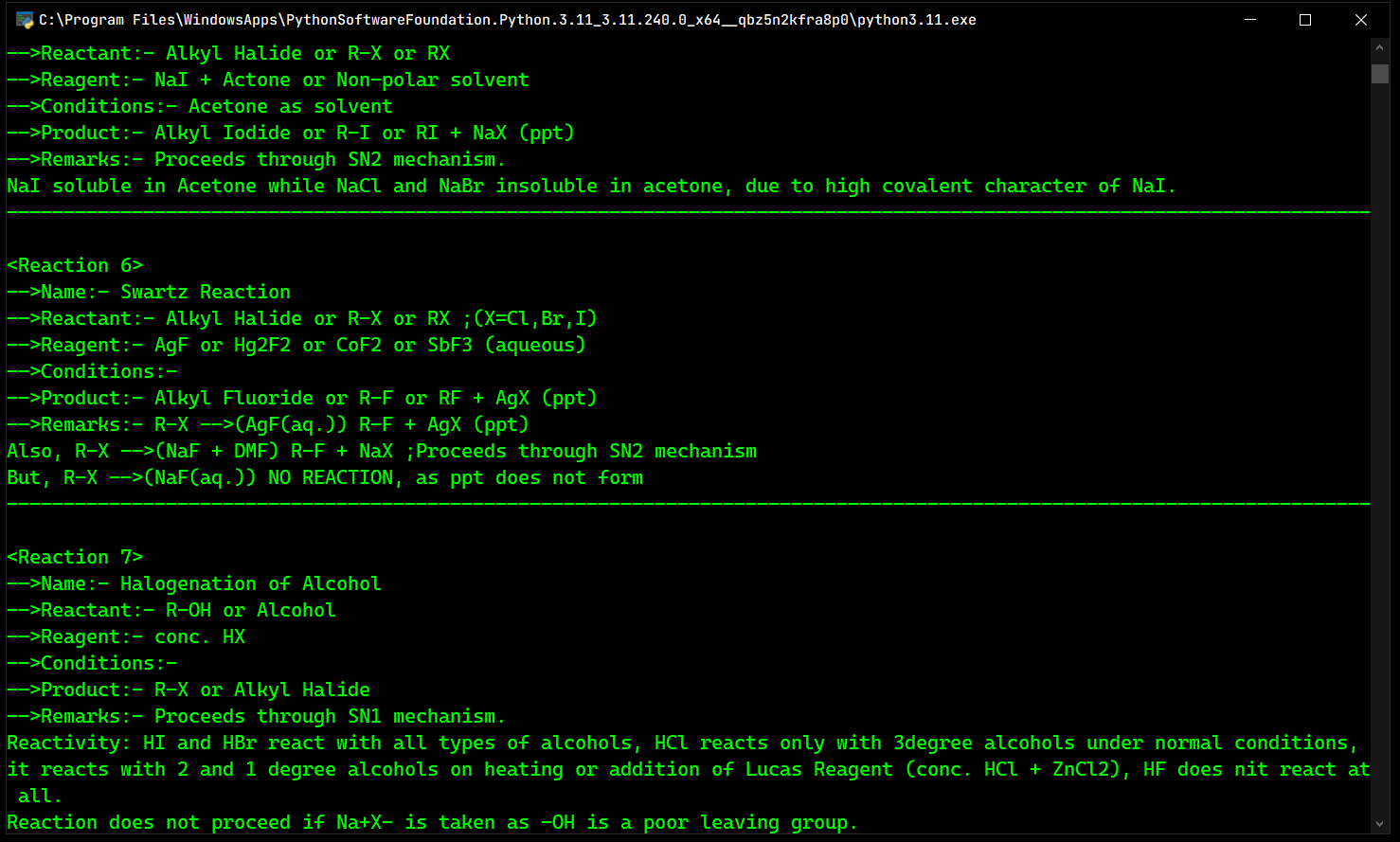
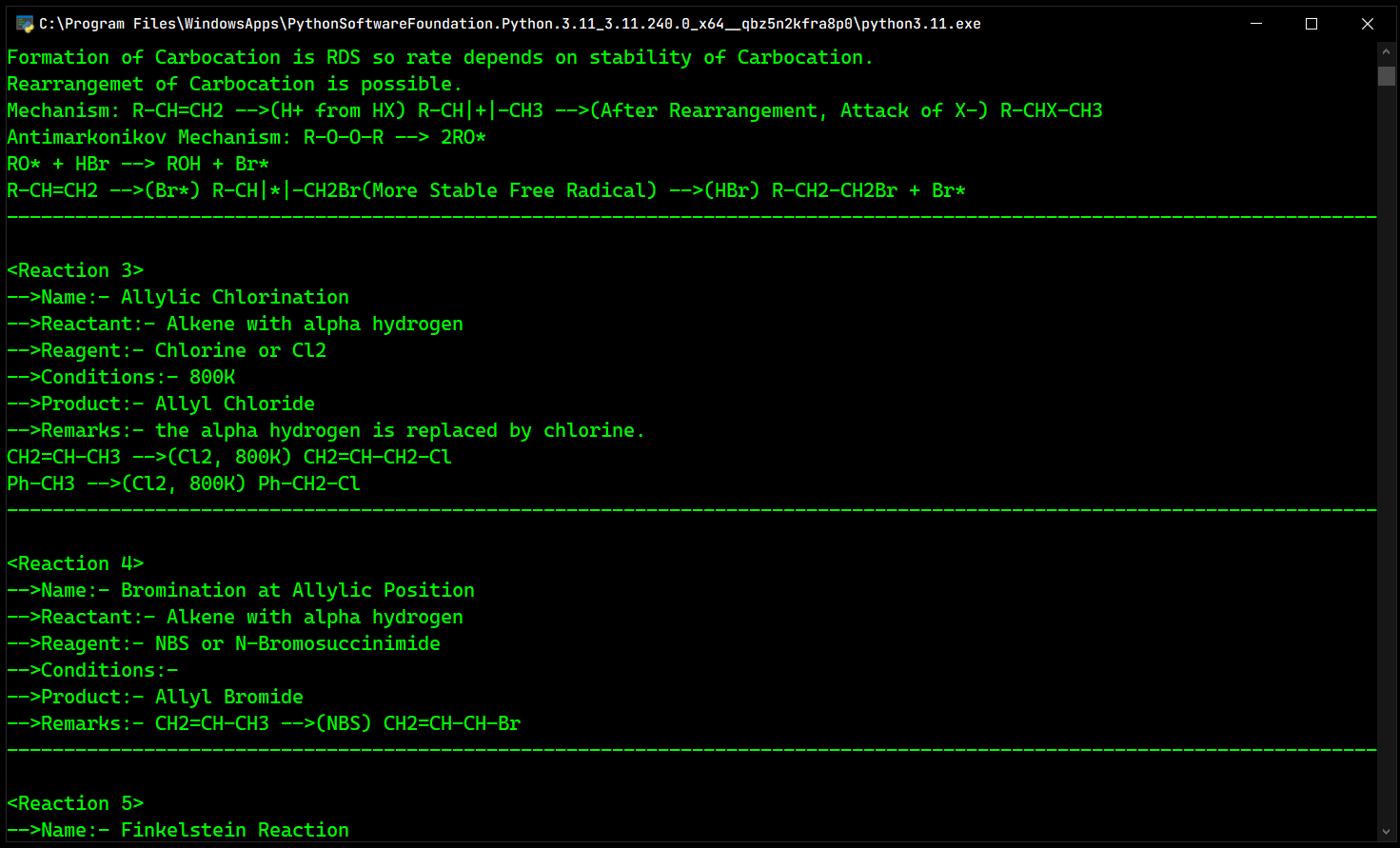
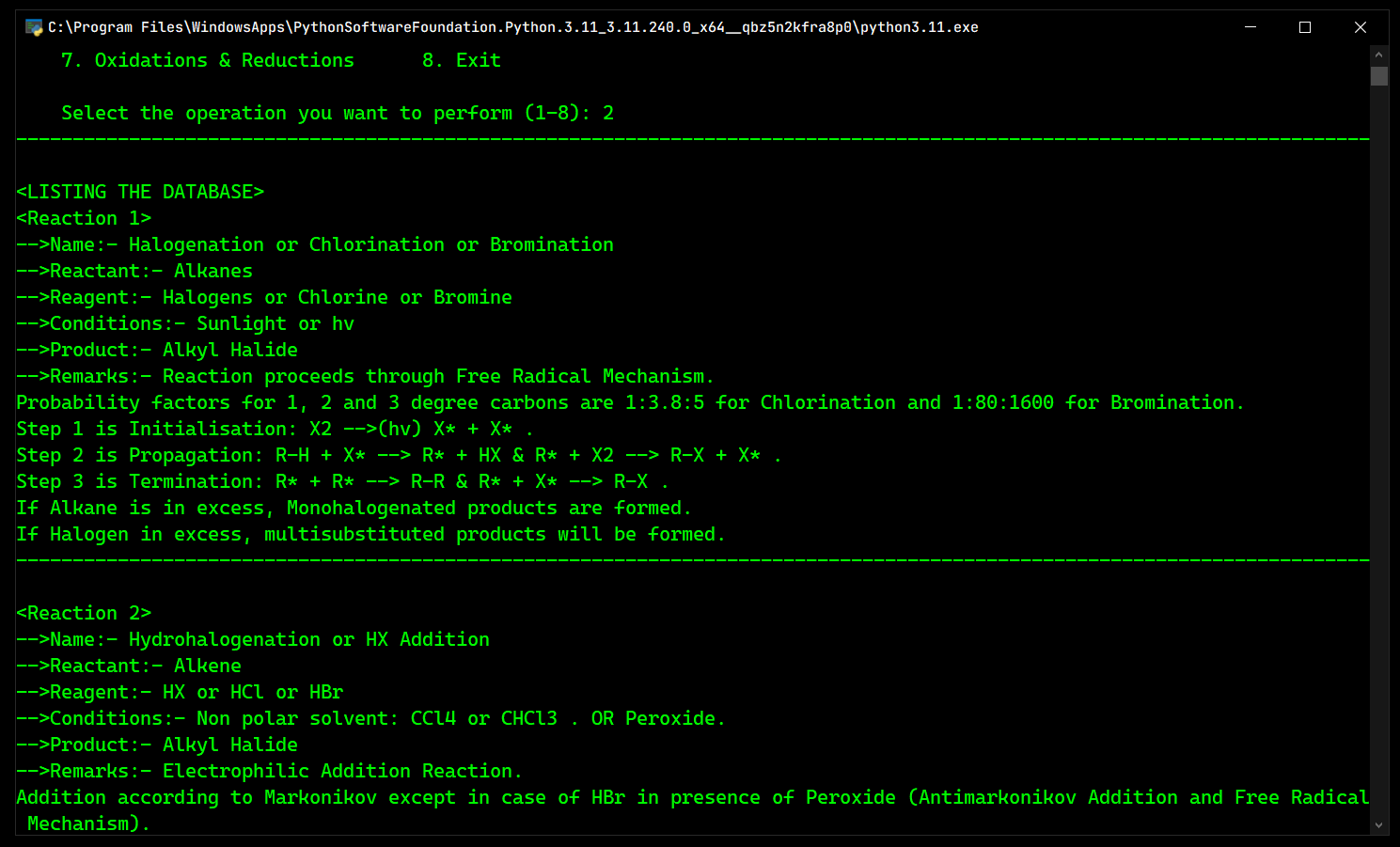
* Start

****

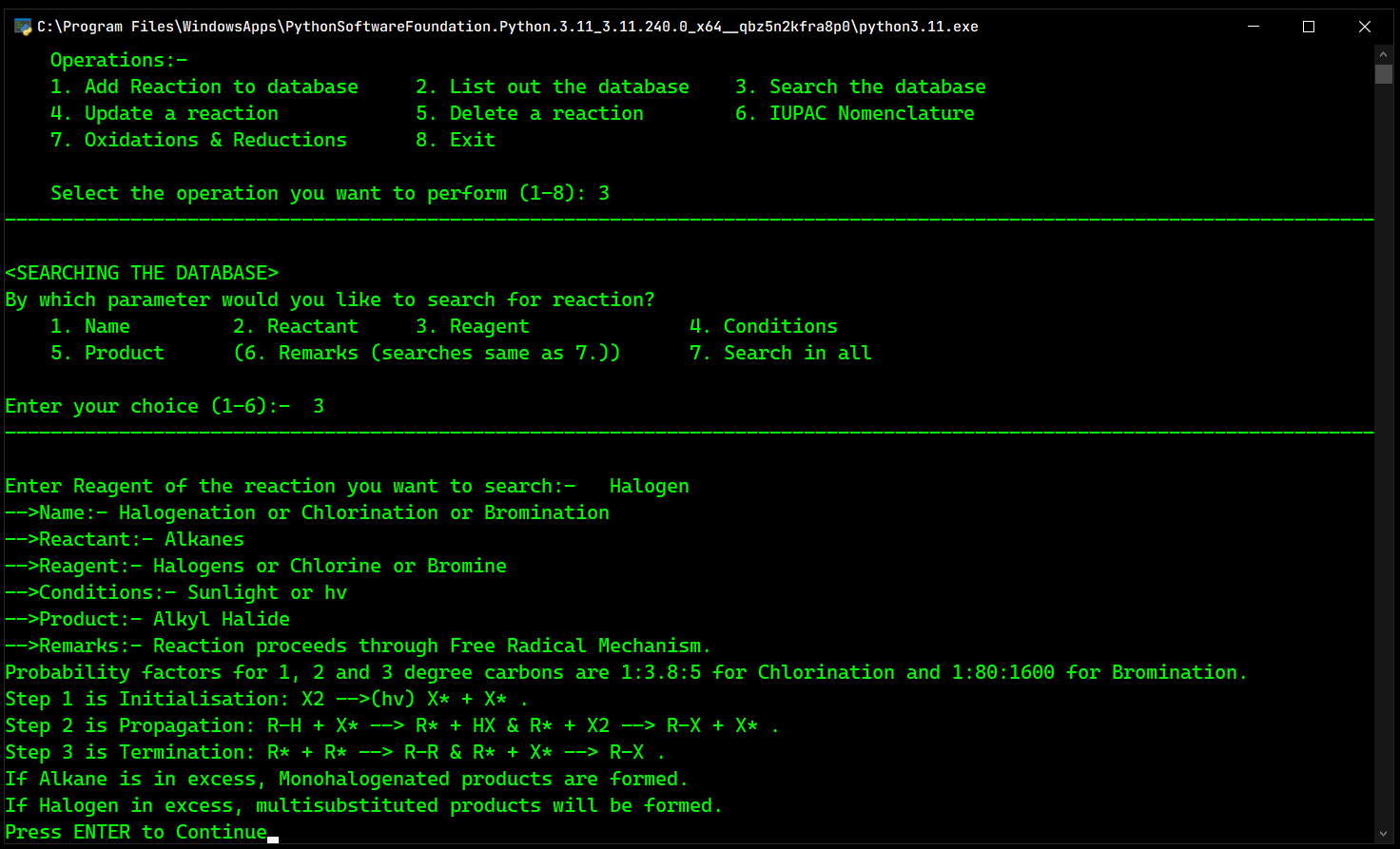
* Operation 1



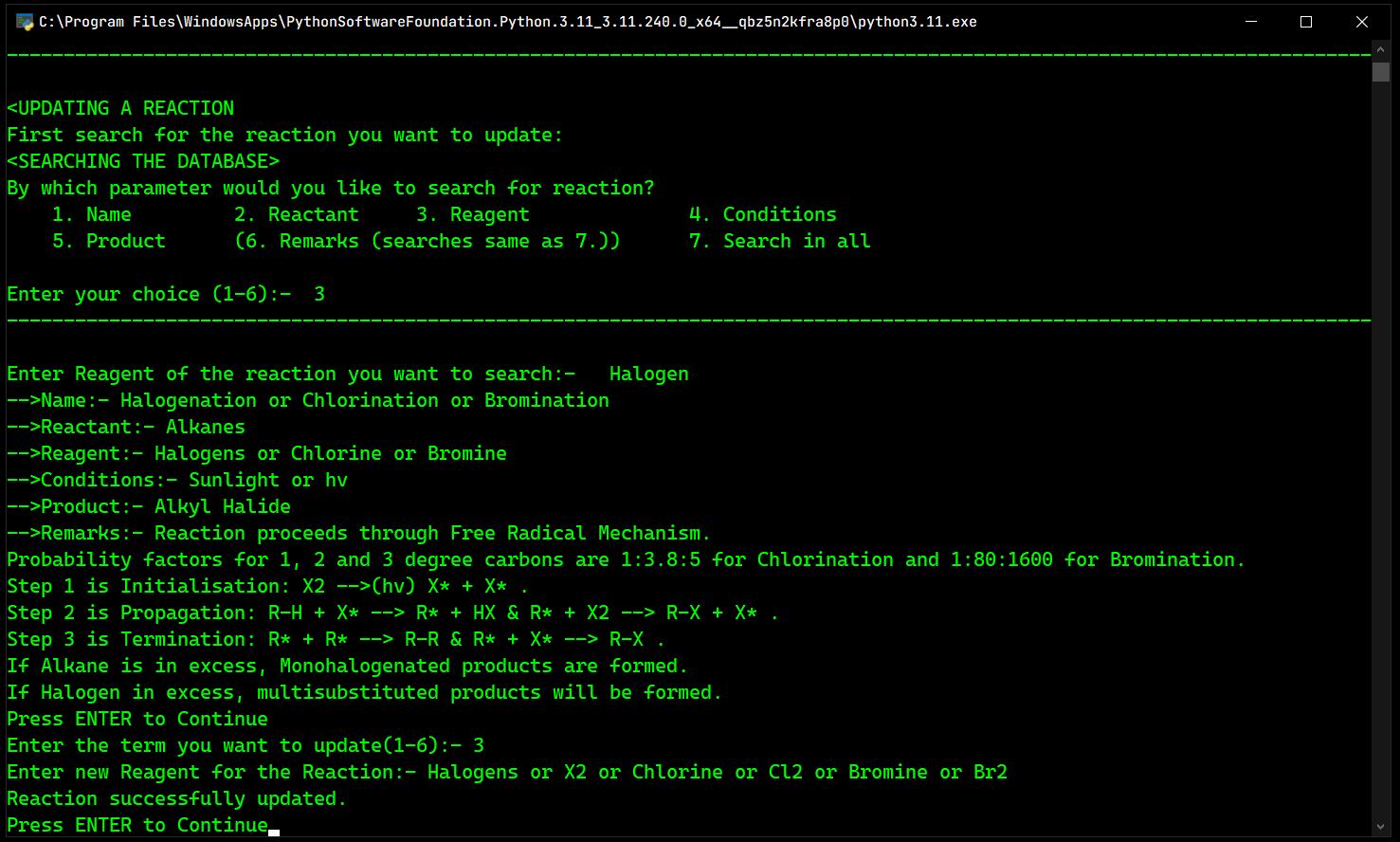
* Operation 2



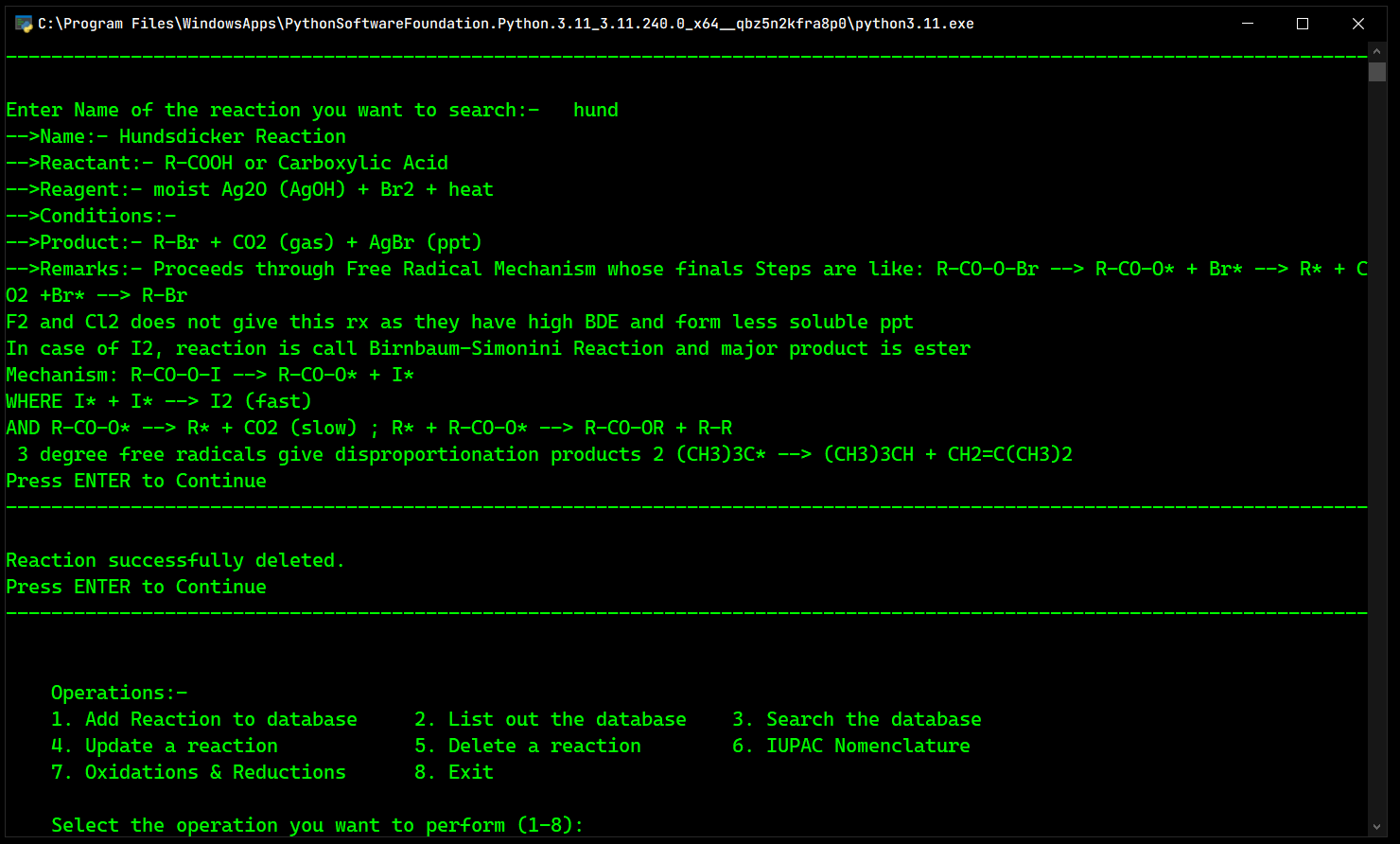
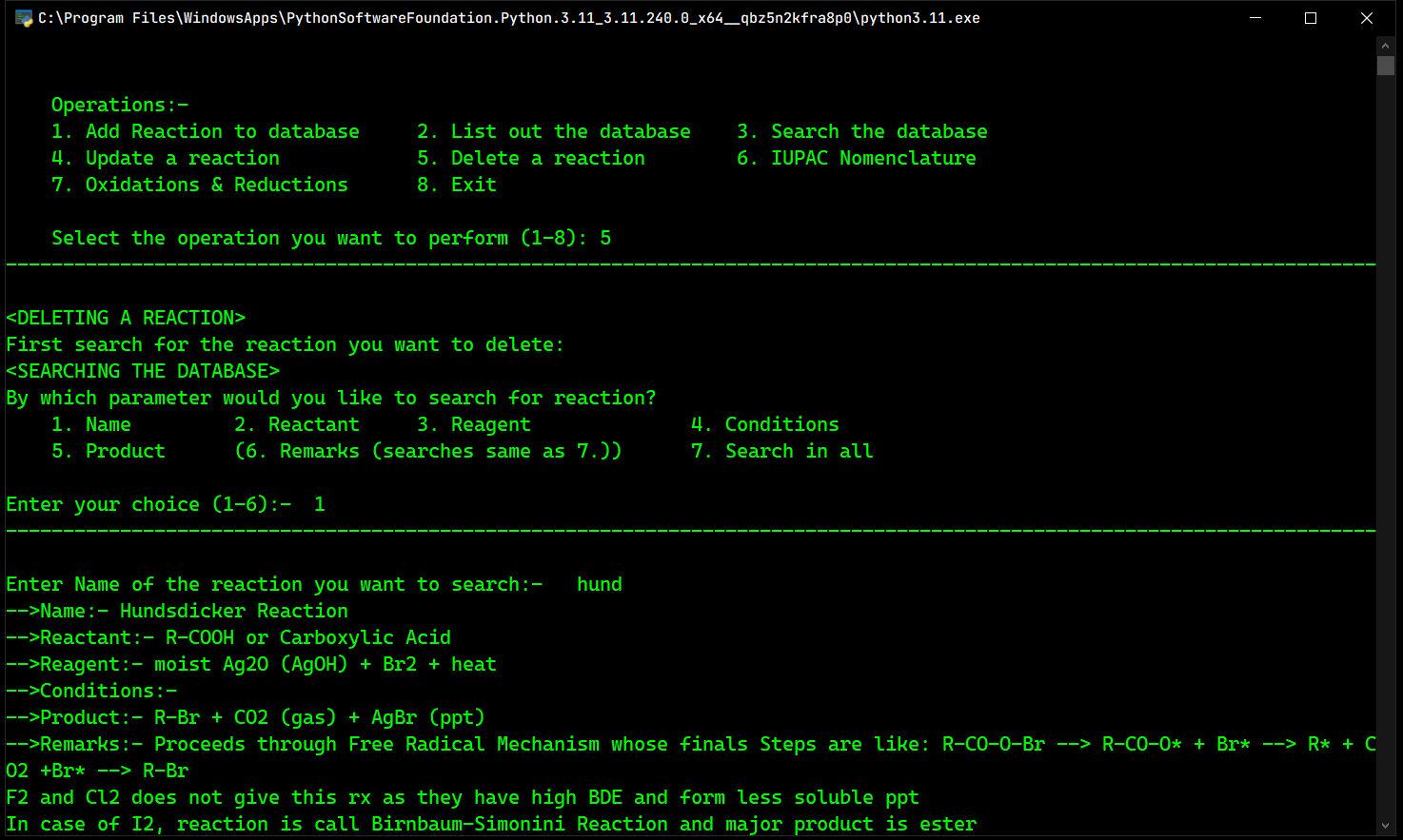
* Operation 3



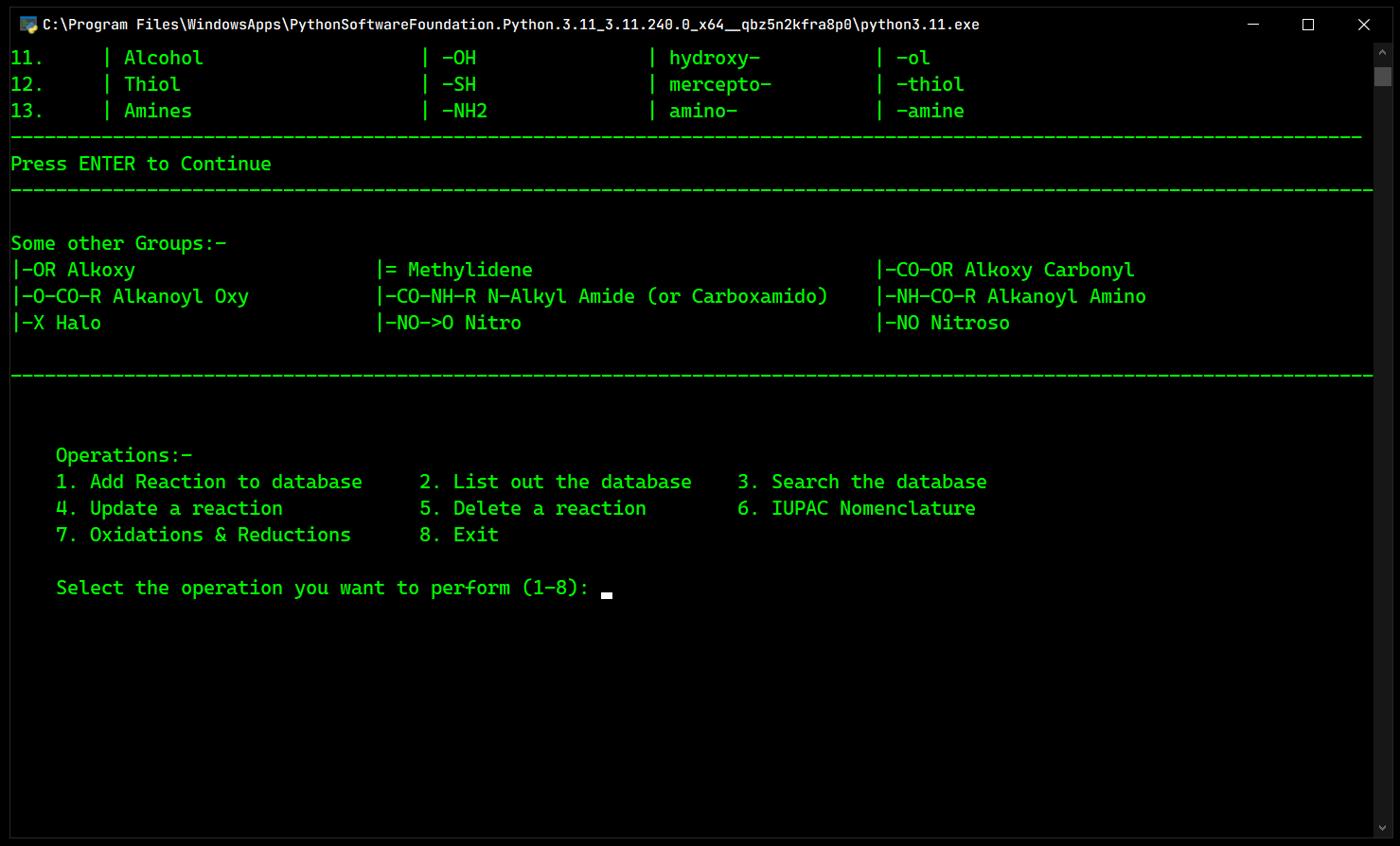
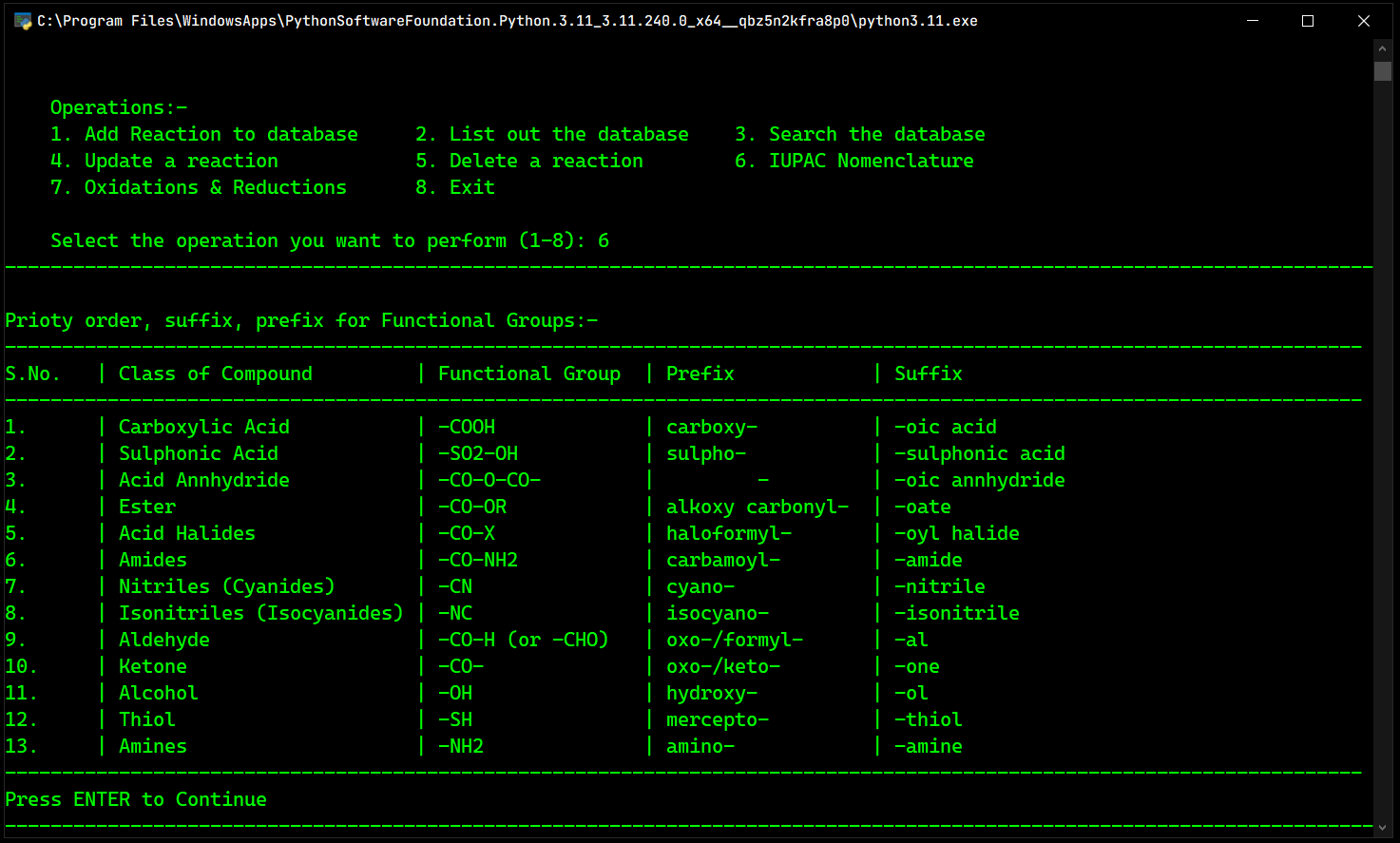
* Operation 4



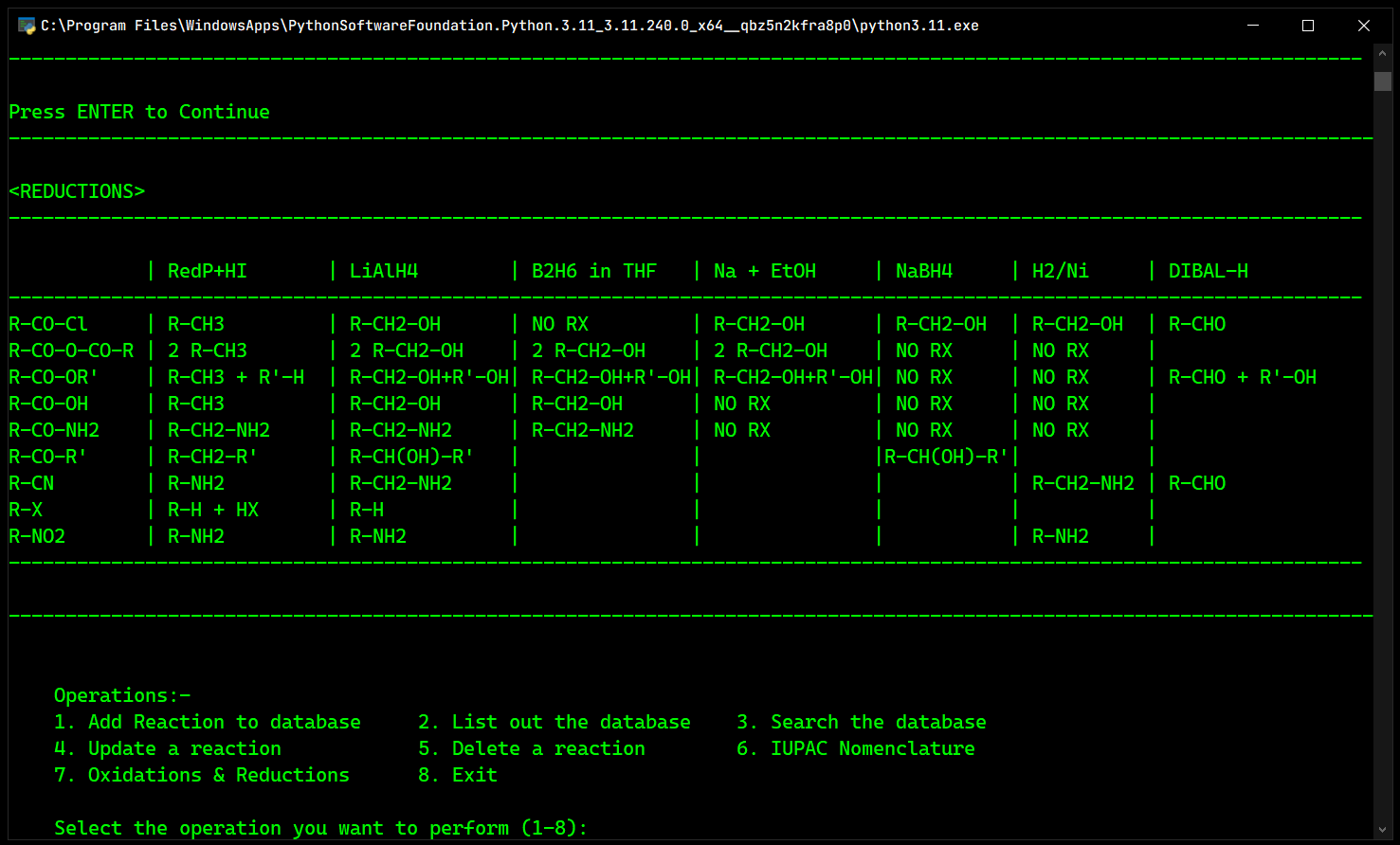
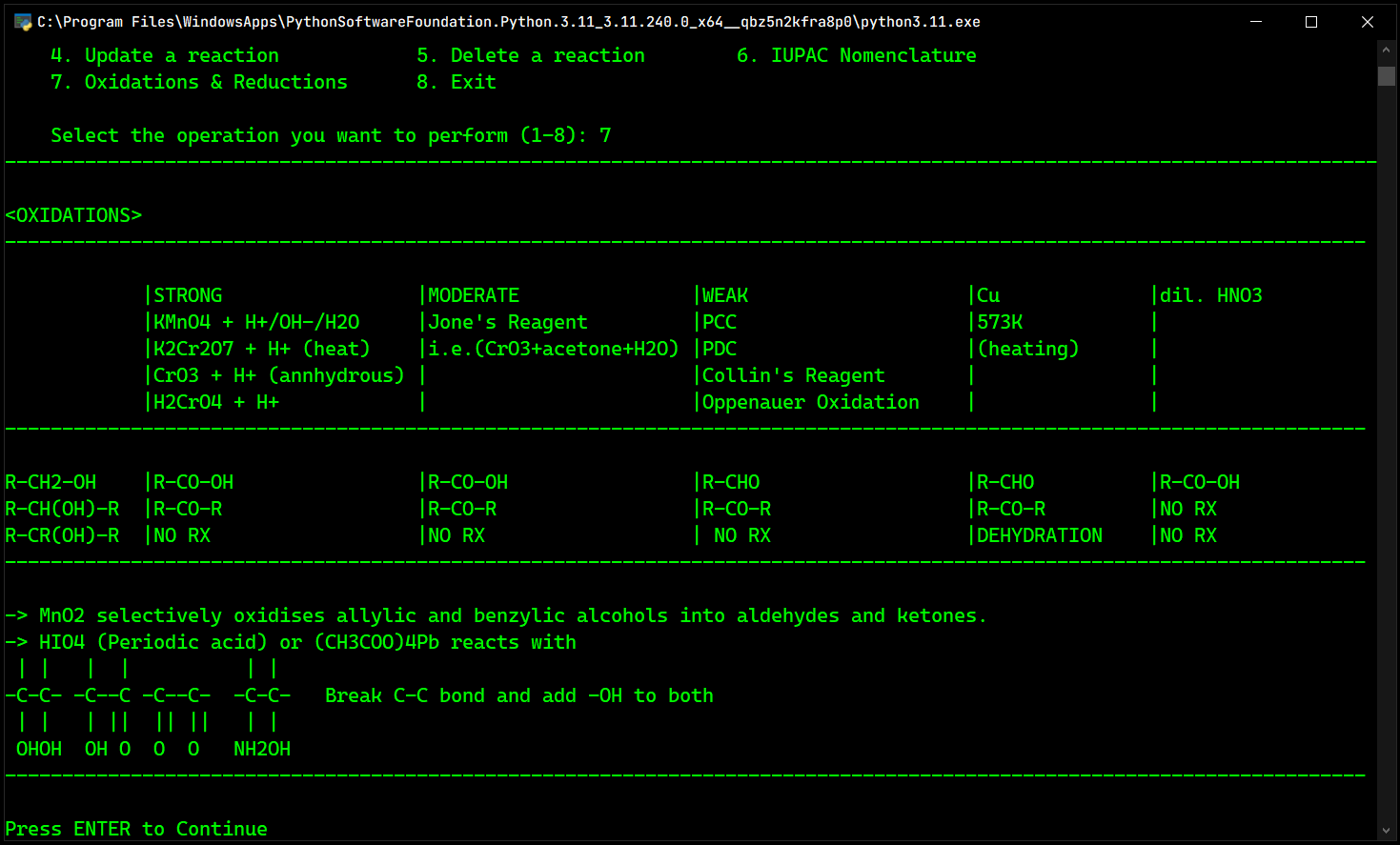
* Operation 5



* Operation 6



* Operation 7



**BIBLIOGRAPHY**

1. Computer Science with python class 12 Sumita Arora
2. <https://stackoverflow.com/>
3. <https://www.geeksforgeeks.org/>
4. <https://dev.mysql.com/doc/refman/8.0/en/>