Practical no.1 IntroductiontopythonInterpreter

Introduction:**Python**isawidelyusedgeneral-purpose,highlevelprogramminglanguage.Itwascreatedby Guidovan Rossumin 1991 andfurtherdeveloped by the Python Software Foundation.Itwas designed with an emphasis on code readability, and its syntax allows programmers to express theirconcepts in fewer lines of code. Python is a programming language that lets you work quickly andintegrate systemsmore efficiently.There are twomajor Pythonversions: Python 2 and Python3.Bothare quitedifferent.

Python isawidelyusedprogramminglanguagethatoffersseveraluniquefeaturesandadvantagescompared to languages like Java and C++. Our Python tutorial thoroughly explains Python basics andadvanced concepts, starting with installation, conditional statements, loops, built-in datastructures, Object-OrientedProgramming, Generators, ExceptionHandling, PythonRegEx,andmanyotherconcepts.Thistutorialisdesignedforbeginnersandworkingprofessionals.

WhatisPython. . .

Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, garbage-collected,andpurelyobject-orientedprogramminglanguagethatsupportsprocedural,object-oriented,andfunctionalprogramming.

0.1 Application:

- WebDevelopment
- Machinelearningandartificialintelligence
- DataScience
- GameDevelopment
- AudioandVideoapplications
- SoftwareDevelopment
- CADApplications
- BusinessApplications
- DesktopGUI
- WebScrapingApplication

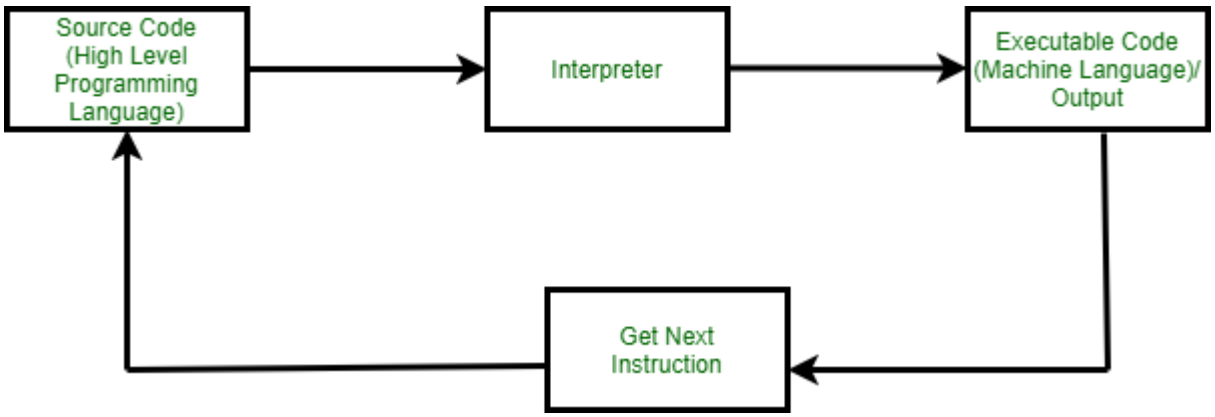
0.2 Featuresofpython:

- Freeandopensource
- Easytocode
- Easytoread
- Objectorientedlanguage
- GUIprogrammingsupport
- High-levellanguage
- Largecommunitysupport
- Easytodebug
- Pythonandportablelanguage
- Pythonisaintegratedlanguage
- Dynamicallytypedlanguage

0.3 Differencebetweencompilerandinterpreter:

S.NO.	Compiler	Interpreter
1.	StepsofProgramming: • ProgramCreation. • Analysisoflanguagebythecompilerandthrowserrorsincaseofanyincorrectstatement. • Incaseofnoerror,theCompilerconvertsthesourcecodetoMachineCode. • Linkingofvariouscodefilesintoarunnable program. • FinallyrunsaProgram.	StepsofProgramming: • ProgramCreation. • LinkingoffilesorgenerationofMachineCodeisnotrequired • Executionofsourcestatementsonebyone.
2.	ThecompilersavestheMachineLanguageinformofMachineCodeondisks.	TheInterpreterdoesnotsavetheMachineLanguage.
3.	CompiledcodesrunfasterthanInterpreter.	InterpretedcodesrunslowerthanCompiler.
4.	Linking-LoadingModelisthebasicworkingmodeloftheCompiler.	TheInterpretationModelisthebasicworkingmodelofthe Interpreter.
5.	Thecompilergeneratesanoutputintheformof(.exe).	Theinterpreterdoesnotgenerate anyoutput.
6.	Any change in the source program after thecompilationrequiresrecompilingtheentirecode.	Anychangeinthesourceprogramduringthetranslationdoesnotreq code.
7.	ErrorsaredisplayedinCompilerafterCompilingtogetheratthecurrentline.	Errorsaredisplayedineverysingleline.
8.	Thecompilercanseecodeupfrontwhichhelps in running the code faster because ofperformingOptimization.	The Interpreter works by line working of Code,that'swhyOptimizationisalittleslowercomparedtoCompil

Python interpreter: Interpreters are the computer program that will convert the source code or an highlevellanguageintointermediatecode(machinelevellanguage).Itisalsocalledtranslatorinprogramming terminology. Interpreters executes each line of statements slowly. This process is calledInterpretation.ForexamplePythonisaninterpretedlanguage,PHP,Ruby,andJavaSc



WorkingofInterpreter

0.4 Pythonversions:

1. Python3.12.4,documentationreleasedon6June2024.
2. Python3.12.3,documentationreleasedon9April2024.
3. Python3.12.2,documentationreleasedon6February2024.
4. Python3.12.1,documentationreleasedon8December2023.
5. Python3.12.0,documentationreleasedon2 October2023.
6. Python3.11.9,documentationreleasedon2April2024.
7. Python3.11.8,documentationreleasedon6February2024.
8. Python3.11.7,documentationreleasedon4December2023.
9. Python3.11.6,documentationreleasedon2 October2023.
10. Python3.11.5,documentationreleasedon24August2023.

0.5 PythonIDE:

- PyCharm
- IDLE
- VisualStudioCode
- Atom
- SublimeText

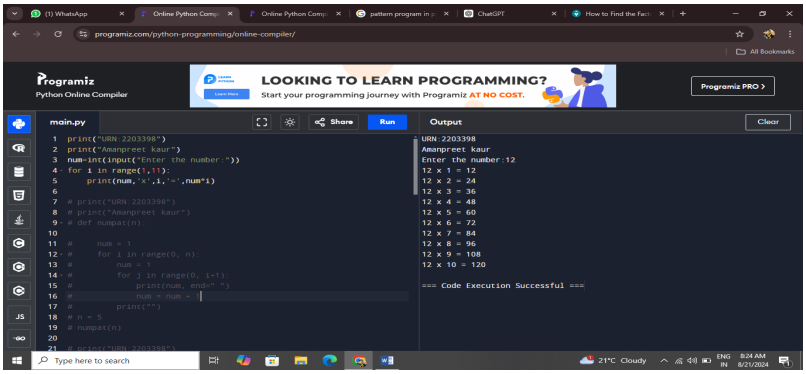
- Spyder
- PyDev

1 Practicalno

Programstoimplement inputoutputandcontrolflowtoolsinpython.

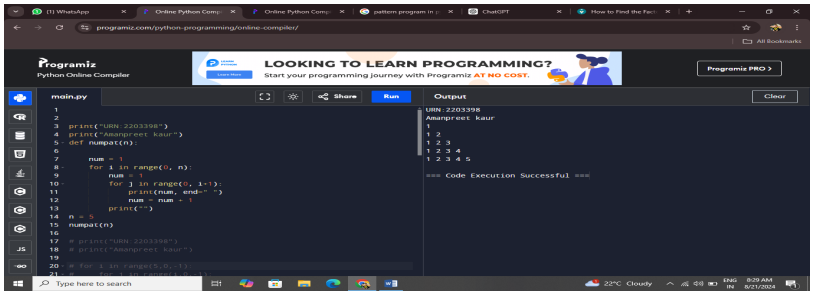
1.1 Multiplicationtableofthenumberenteredbytheuser

Output:

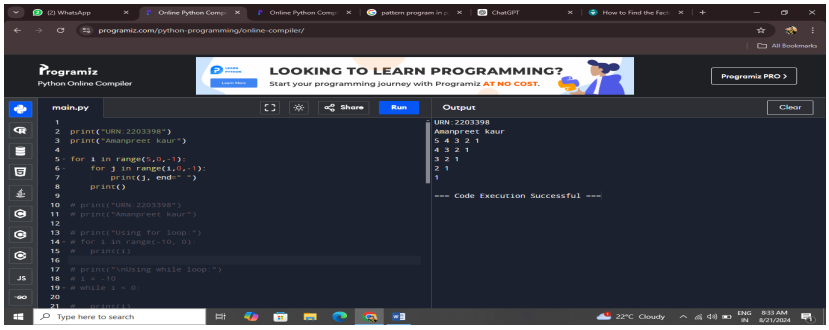


1.2 Printthepattern:

Output:



Output:



Practical2:

Programstoimplement inputoutputandcontrolflowtoolsinpython.

1.3 Multiplicationtableofthenumberenteredbytheuser

```
print("URN:2203398")
print("Amanpreet kaur")
num=int(input("Enter the number:"))
for i in range(1,11):
   print(num,'x',i,'=',num*i)
```

1.4 Printthefollowingpattern:

	1
	1 2
	123
	1 23 4
	1 23 45

```
print("URN:2203398")
print("Amanpreet kaur")
def numpat(n):

   num = 1
   for i in range(0, n):
      num = 1
      for j in range(0, i+1):
```

```
            print(num, end=" ")
            num = num + 1
        print("")
n = 5
numpat(n)
```

1.5 43 21

4 32 1

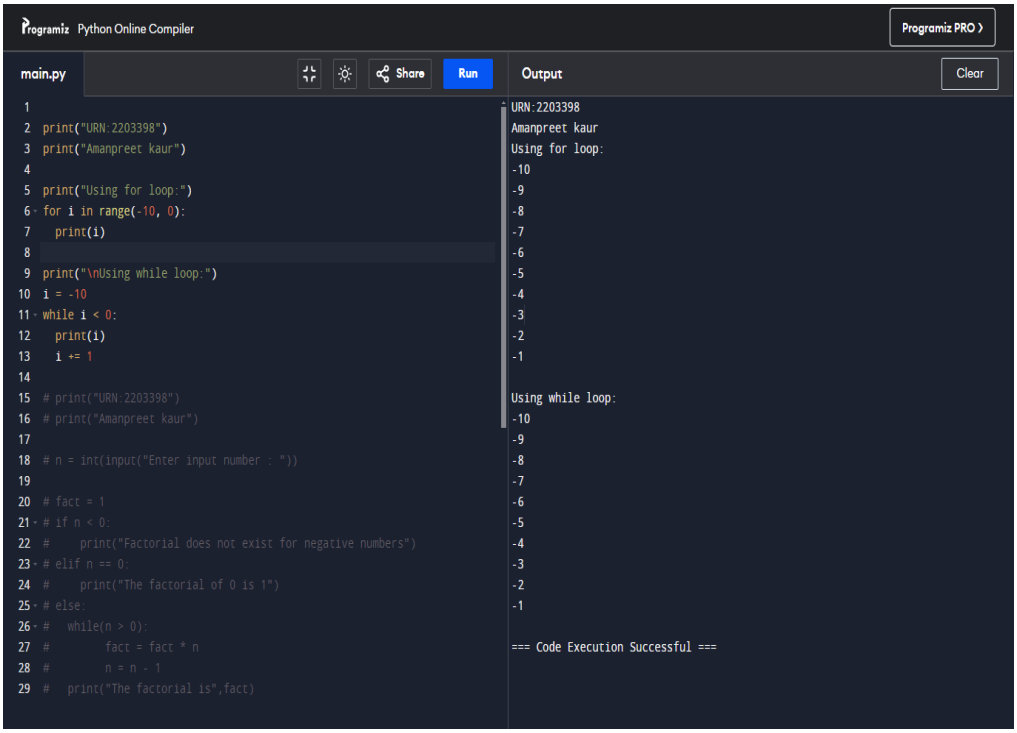
1.6 21

2 1
1

```
print("URN:2203398")
print("Amanpreet kaur")

for i in range(5,0,-1):
    for j in range(i,0,-1):
        print(j, end=" ")
    print()
```

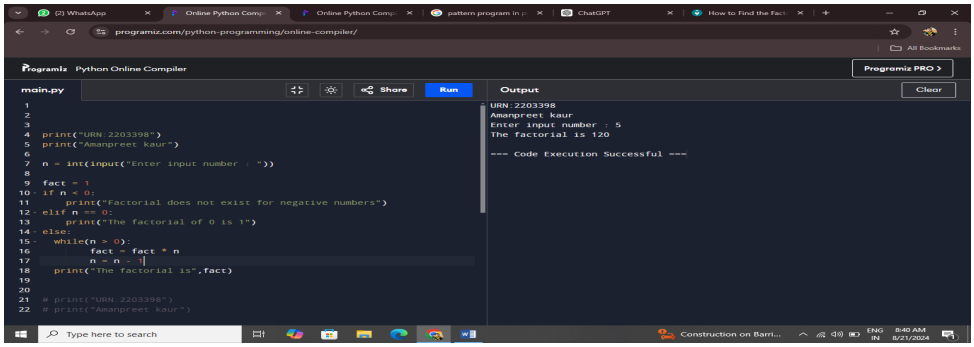
1.7 Displaynumberfrom -10to -1 usingforloop and whileloop.

Output:



1.8 Findafactorial ofnumberenteredbyuser.

Output:



1.9 Displaynumberfrom -10to -1 usingforloop and whileloop.

```
print("URN:2203398")
print("Amanpreet kaur")

print("Using for loop:")
for i in range(-10, 0):
  print(i)
```

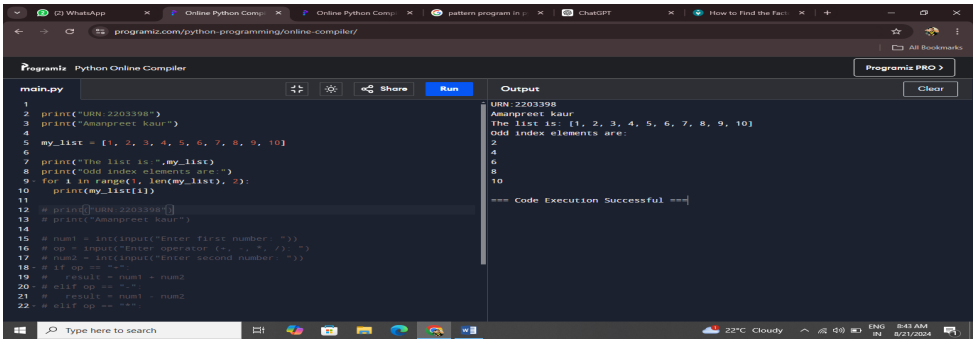
```
print("\nUsing while loop:")
i = -10
while i < 0:
  print(i)
  i += 1
```

1.10 Findafactorialofnumberenteredbyuser. Findafactorialofnumberenteredbyuser

```
print("URN:2203398")
print("Amanpreet kaur")

n = int(input("Enter input number : "))

fact = 1
if n < 0:
    print("Factorial does not exist for negative numbers")
elif n == 0:
    print("The factorial of 0 is 1")
else:
  while(n > 0):
        fact = fact * n
        n = n - 1
  print("The factorial is",fact)
```

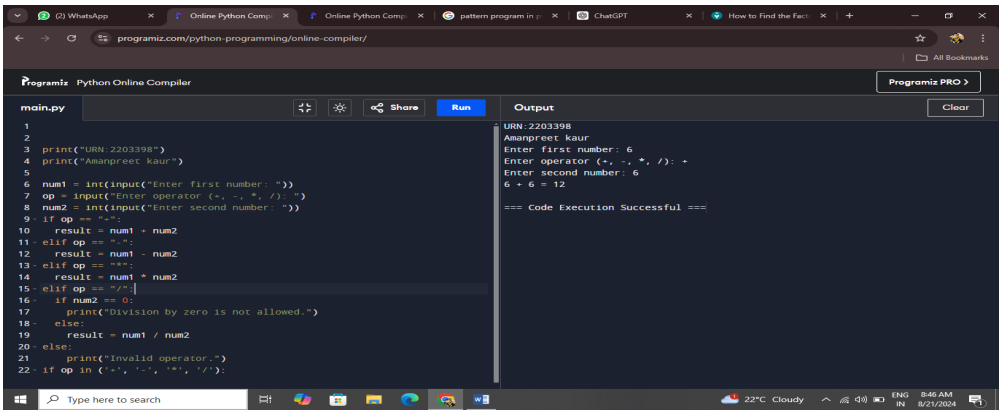
1.11 Usingalooptodisplayelementsfrom a givenlistpresentat oddindexposition.

Output:



1.12 Makeasimplecalculator. Makeasimplecalculator

Output:



1.13 Usingalooptodisplayelementsfrom a givenlistpresentat odd indexposition.

```
 print("URN:2203398")
print("Amanpreet kaur")

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print("The list is:",my_list)
print("Odd index elements are:")
for i in range(1, len(my_list), 2):
 print(my_list[i])
```

1.14 Makeasimplecalculator. Makeasimplecalculator

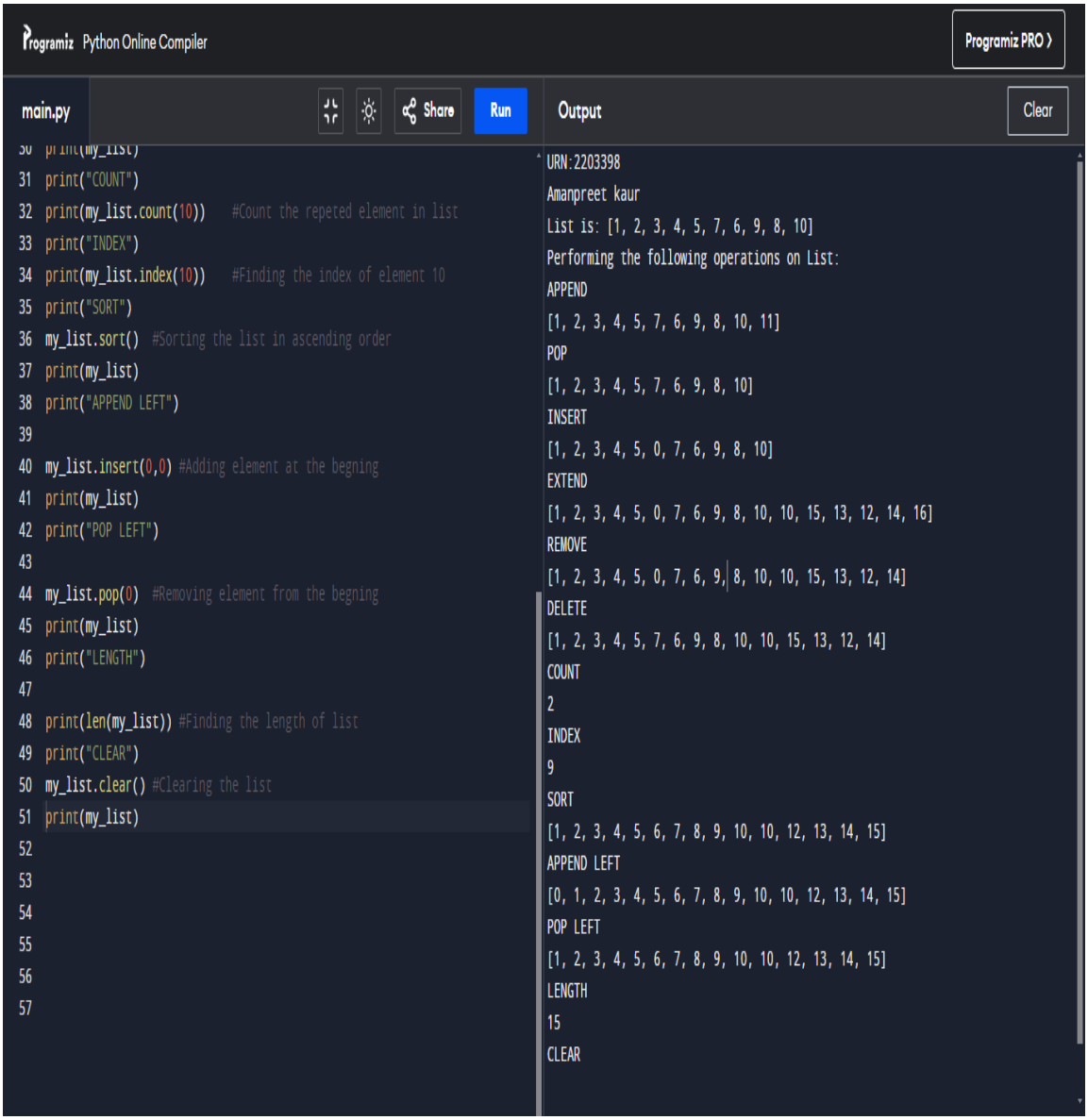
```
print("URN:2203398")
print("Amanpreet kaur")

num1 = int(input("Enter first number: "))
op = input("Enter operator (+, -, ∗, /): ")
num2 = int(input("Enter second number: "))
if op == "+":
 result = num1 + num2
elif op == "-":
 result = num1 - num2
elif op == "∗":
 result = num1 ∗ num2
```

```
elif op == "/":
  if num2 == 0:
    print("Division by zero is not allowed.")
  else:
    result = num1 / num2
else:
    print("Invalid operator.")
if op in ('+', '-', '*', '/'):
  print(num1, op, num2, "=", result)
```

2 Practical 3:Programtoimplementdifferentdatastructurein Python.

2.1 List

Output:



3 Practical no. 3Programtoimplementdifferentdatastructureinpython.List:

PythonListsarejustlikethearrays,declaredinotherlanguageswhichareanorderedcollectionofdata.Itisveryflexibleastheitemsinalist

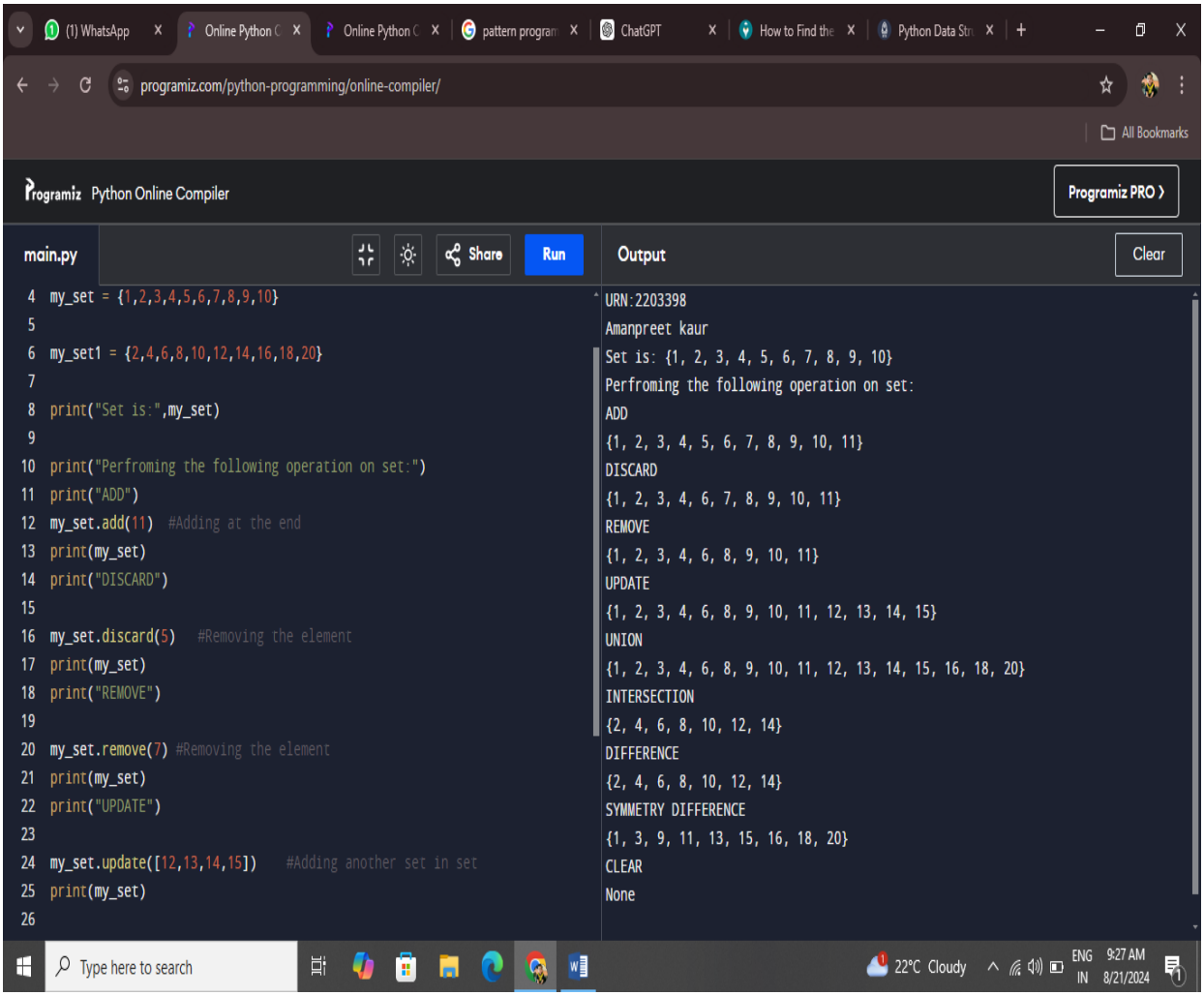
print("URN:2203398")
print("Amanpreet kaur")

my_list = [1,2,3,4,5,7,6,9,8,10]#Create list or using list()print("Listis:",my_list)
print("Performingthefollowingoperationson List:")print("APPEND")
my_list.append(11) #Addingattheendprint(my_list)
print("POP")

```
my_list.pop()    #Removinglastelementprint(my_list)
print("INSERT")

my_list.insert(5,0)    #Inserting0atindex5print(my_list)
print("EXTEND")

my_list.extend([10,15,13,12,14,16])    #Addinglisttothelistprint(my_list)
print("REMOVE")

my_list.remove(16)    #Removing16fromthelistprint(my_list)
print("DELETE")

delmy_list[5]    #Deletingelement fromindex5

print(my_list)print("COUNT")
print(my_list.count(10))    #Counttherepetedelementinlistprint("INDEX")
print(my_list.index(10))    #Findingtheindexof element10print("SORT")
my_list.sort()    #Sortingthelistinascendingorderprint(my_list)
print("APPENDLEFT")

my_list.insert(0,0)#Addingelement at thebegningprint(my_list)
print("POP LEFT")

my_list.pop(0)    #Removing elementfromthebegningprint(my_list)
print("LENGTH")

print(len(my_list))    #Findingthelengthoflistprint("CLEAR")
my_list.clear()    #Clearingthelistprint(my_list)
```

3.1 Set:

Output:



3.2 Set:

PythonSetisanunorderedcollectionofdatathatismutable anddoesnotallowanyduplicateelement.Setsare basicallyusedtoin-cludemembershiptestingandeliminatingduplicateentries.ThedatastructureusedinthisisHashing,apopulartechniquetoperformins

print("URN:2203398")
print("Amanpreet kaur")

my_set ={1,2,3,4,5,6,7,8,9,10}

my_set1= {2,4,6,8,10,12,14,16,18,20}

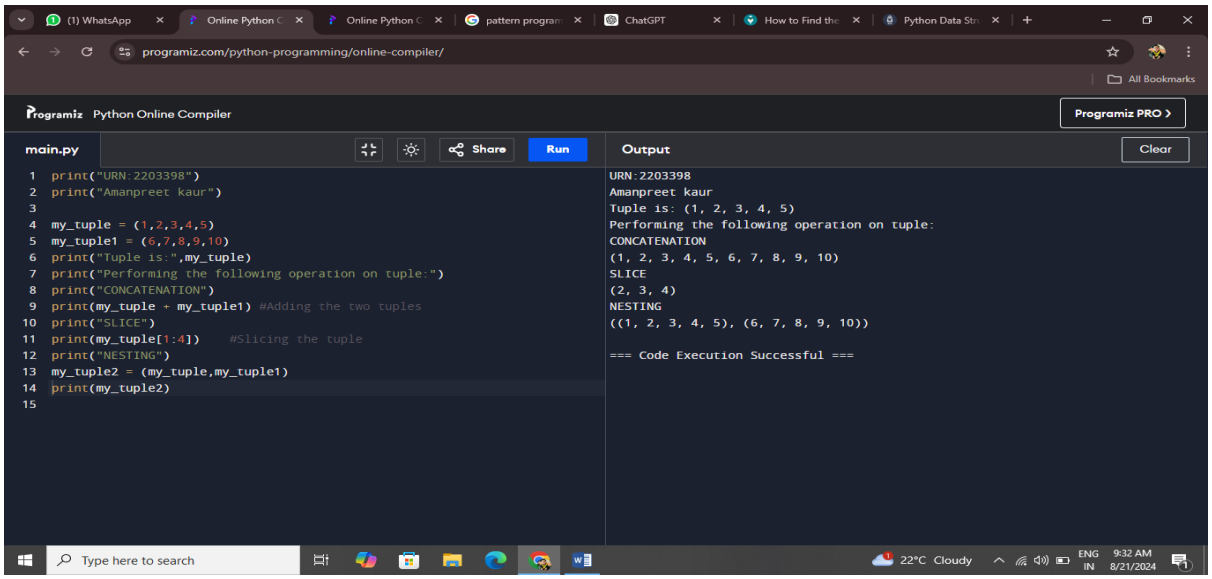
print("Setis:",my_set)

print("Perfromingthefollowingoperationonset:")print("ADD")
my_set.add(11) #Addingattheendprint(my_set)

```
print("DISCARD")

my_set.discard(5)    #Removingtheelementprint(my_set)
print("REMOVE")

my_set.remove(7)#Removingtheelementprint(my_set)
print("UPDATE")

my_set.update([12,13,14,15])    #Addinganothersetinsetprint(my_set)
print("UNION")

print(my_set.union(my_set1))    #Unionoftwosetsprint("INTERSECTION")
print(my_set.intersection(my_set1))    #Intersectionoftwosetsprint("DIFFERENCE")

print(my_set.intersection(my_set1))    #Intersectionoftwosetsprint("SYMMETRYDIFFERENCE")
print(my_set.symmetric_difference(my_set1))#Differentelementsprint("CLEAR")
print(my_set.clear())    #Clearingtheset
```

3.3 Tuple

Output:



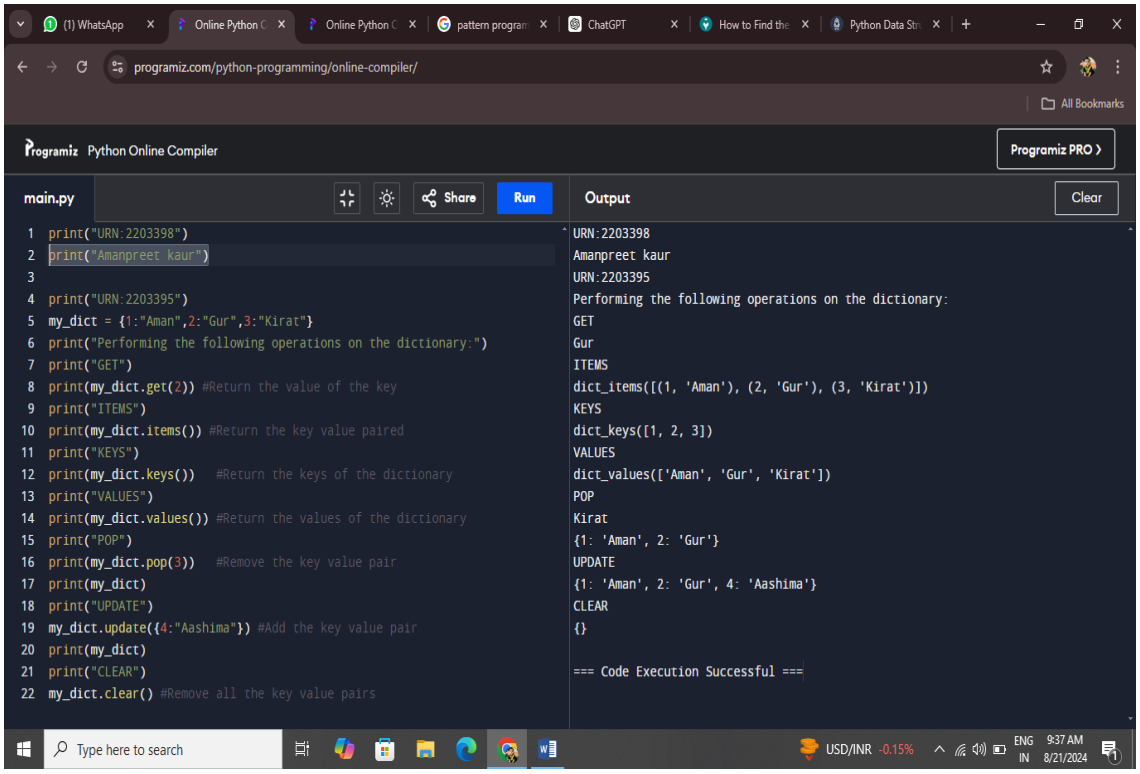
3.4 Tuple:

PythonTupleisacollectionofPythonobjectsmuchlikealistbutTuplesareimmutableinnaturei.e.theelementsinthetuplecannotbe addedorremovedoncecreated.JustlikeaList,aTuplecanalsocontainelementsofvarioustypes.

```
print("URN:2203398")
print("Amanpreet kaur")

my_tuple = (1,2,3,4,5)
my_tuple1 = (6,7,8,9,10)
print("Tuple is:",my_tuple)
print("Performing the following operation on tuple:")
print("CONCATENATION")
print(my_tuple + my_tuple1)    #Adding the two tuples
print("SLICE")
print(my_tuple[1:4])    #Slicing the tuple
print("NESTING")
my_tuple2 = (my_tuple,my_tuple1)
print(my_tuple2)
```

3.5 Dictionary

Output:



Dictionary:
Runningsetup.pyinstallfornovas
Successfullyinstallednovas-3.1.1.3numpy-1.9.2requests-2.7.0

Creating and Using Your Own Packages:

- **Creating a Package**: Organize your Python code into modules and then bundle them into a package by including a setup.py file that defines the package metadata.
- **Publishing to PyPI**: Use tools like twine to upload your package to PyPI, making it available for others to install.