

# Practical no.1 Introduction to python Interpreter

**Introduction:** Python is a widely used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: Python 2 and Python 3. Both are quite different.

Python is a widely used programming language that offers several unique features and advantages compared to languages like Java and C++. Our Python tutorial thoroughly explains Python basics and advanced concepts, starting with installation, conditional statements, loops, built-in data structures, Object-Oriented Programming, Generators, Exception Handling, Python RegEx, and many other concepts. This tutorial is designed for beginners and working professionals.

*What is Python... .*

Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, garbage-collected, and purely object-oriented programming language that supports procedural, object-oriented, and functional programming.

## 0.1 Application:

- Web Development
- Machine learning and artificial intelligence
- Data Science
- Game Development
- Audio and Video applications
- Software Development
- CAD Applications
- Business Applications
- Desktop GUI
- Web Scraping Application

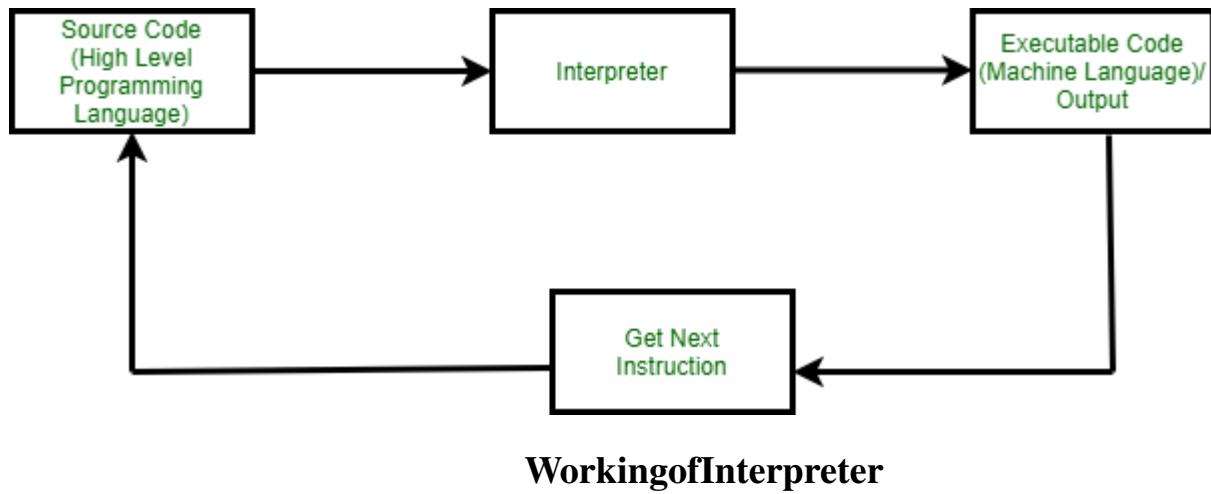
## **0.2 Featuresofpython:**

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

### 0.3 Difference between compiler and interpreter:

S.NO.	Compiler	Interpreter
1.	<p>StepsofProgramming:</p> <ul style="list-style-type: none"> <li>• ProgramCreation.</li> <li>• Analysisoflanguagebythecompilerandthrowserrors.<del>inasmuchasiftheexecutionstatementofMachineCodeisnotrequired</del></li> <li>• Incaseofnoerror,thecompilerconvertsthesourcecodeto<del>MachineCode</del>sourcestatementsonebyone.</li> <li>• Linkingofvariouscodefilesintoarunnableprogram.</li> <li>• FinallyrunsaProgram.</li> </ul>	<p>StepsofProgramming:</p> <ul style="list-style-type: none"> <li>• ProgramCreation.</li> </ul>
2.	Thecompiler savestheMachineLanguageinformofMac	<del>TheInterpreter</del> does not save the Machine Language.
3.	Compiledcodes run faster than Interpreter.	Interpretedcodes run slower than Compiler.
4.	Linking-Loading Model is the basic working model of the Compiler.	<del>Code</del> Interpretation Model is the basic working model of the Interpreter.
5.	Thecompiler generates an output in the form of (.exe).	The interpreter does not generate any output.
6.	Any change in the source program after the compilation requires recompiling the entire code.	Any change in the source program during the translation does not require recompiling the code.
7.	Errors are displayed in Compiler after Compiling together.	<del>Error</del> are displayed in every single line.
8.	The compiler can see code upfront which helps in running the code faster because of performing Optimization.	The Interpreter works by line working of Code, that's why Optimization is a little slower compared to Compiler.

**Python interpreter:** Interpreters are the computer program that will convert the source code or an high level language into intermediate code (machine level language). It is also called translator in programming terminology. Interpreters executes each line of statements slowly. This process is called Interpretation. For example Python is an interpreted language, PHP, Ruby, and Java are compiled languages.



#### **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

Programs to implement input output and control flow tools in python.

## 1.1 Multiplication table of the number entered by the user

Output:

```

main.py
1 print("UN 2203398")
2 print("Amanpreet kaur")
3 num = int(input("Enter the number:"))
4 for i in range(1,11):
5     print(num,'x',i,'=',num*i)
6
7 # print("UN 2203398")
8 # print("Amanpreet kaur")
9 # def mapat(n):
10 #
11 #     num = 1
12 #     for i in range(0, n):
13 #         num = 1
14 #         for j in range(0, i+1):
15 #             print(num, end=" ")
16 #             num = num + 1
17 #         print()
18 #     n -= 5
19 # mapat(n)
20

```

Output

```

UN 2203398
Amanpreet kaur
Enter the number: 12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
== Code Execution Successful ==

```

## 1.2 Print the pattern:

Output:

```

main.py
1
2
3 print("UN 2203398")
4 print("Amanpreet kaur")
5 def mapat(n):
6
7     num = 1
8     for i in range(0, n):
9         num = 1
10        for j in range(0, i+1):
11            print(num, end=" ")
12            num = num + 1
13        print()
14
15 mapat(5)
16
17 # print("UN 2203398")
18 # print("Amanpreet kaur")
19 # for i in range(5,0,-1):
20 #     mapat(i)

```

Output

```

UN 2203398
Amanpreet kaur
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
== Code Execution Successful ==

```

Output:

The screenshot shows a browser window with multiple tabs open. The active tab is 'Programiz Python Online Compiler' at [programiz.com/python-programming/online-compiler/](https://programiz.com/python-programming/online-compiler/). The code in the editor is:

```

1 print("URN:2203398")
2 print("Amanpreet kaur")
3
4 for i in range(1,11):
5     for j in range(i,11):
6         print(j, end=" ")
7     print()
8
9
10 # print("URN 2203398")
11 # print("Amanpreet kaur")
12 # print("using for loop:")
13 # for i in range(10,0,-1):
14 #     for j in range(10,0,-1):
15 #         print(j)
16 #
17 # print("using while loop:")
18 # i = 10
19 # while i > 0:
20 #     for j in range(10,0,-1):
21 #         print(j)

```

The output window shows the printed numbers:

```

URN:2203398
Amanpreet kaur
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1

```

Below the output, a message reads: 'Code Execution Successful ---'

Practical2:

**Programs to implement input output and control flow tools in python.**

### 1.3 Multiplication table of the number entered by the user

```

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 Print the following pattern:

	<b>1</b>
	<b>1 2</b>
	<b>123</b>
	<b>1 23 4</b>
	<b>1 23 45</b>

```

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 Display number from -10 to -1 using for loop and while loop.**

Output:

The screenshot shows the Programiz Python Online Compiler interface. The code in main.py prints "URN:2203398", "Amanpreet kaur", and then uses a for loop to print numbers from -10 to 0. It then prints "Using while loop:" and uses a while loop to print the same range of numbers. The output window shows the printed statements and the results of both loops.

```

1 print("URN:2203398")
2 print("Amanpreet kaur")
3
4 print("Using for loop:")
5 for i in range(-10, 0):
6     print(i)
7
8 print("\nUsing while loop:")
9 i = -10
10 while i < 0:
11     print(i)
12     i += 1
13
14
15 # print("URN:2203398")
16 # print("Amanpreet kaur")
17
18 # n = int(input("Enter input number : "))
19
20 # fact = 1
21 # if n < 0:
22 #     print("Factorial does not exist for negative numbers")
23 # elif n == 0:
24 #     print("The factorial of 0 is 1")
25 # else:
26 #     while(n > 0):
27 #         fact = fact * n
28 #         n = n - 1
29 #     print("The factorial is",fact)

```

Output:

```

URN:2203398
Amanpreet kaur
Using for loop:
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1

Using while loop:
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1

```

==== Code Execution Successful ===

## 1.8 Find a factorial of number entered by user.

Output:

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Programiz Python Online Compiler". The code in main.py prompts for an input number, checks for non-negative values, and calculates the factorial using a while loop. The output shows the input number and the calculated factorial.

```

1
2
3
4 print("URN:2203398")
5 print("Amanpreet kaur")
6
7 n = int(input("Enter input number : "))
8
9 fact = 1
10 if n < 0:
11     print("Factorial does not exist for negative numbers")
12 elif n == 0:
13     print("The factorial of 0 is 1")
14 else:
15     while(n > 0):
16         fact = fact * n
17         n = n - 1
18     print("The factorial is",fact)
19
20
21 # print("URN:2203398")
22 # print("Amanpreet kaur")

```

Output:

```

URN:2203398
Amanpreet kaur
Enter input number : 5
The factorial is 120

```

==== Code Execution Successful ===

## 1.9 Display number from -10 to -1 using for loop and while loop.

```

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

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

print("Using while loop:")
i = -10
while i >= 0:
    print(i)
    i -= 1

```

```

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

```

## 1.10 Find a factorial of number entered by user. Find a factorial of number entered by user

```

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 Using a loop to display elements from a given list present at odd index position.

Output:

```

main.py
1
2 print("URN: 2203398")
3 print("Amanpreet kaur")
4
5 my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
6
7 print("The list is-",my_list)
8 print("Odd index elements are:")
9 for i in range(1, len(my_list), 2):
10     print(my_list[i])
11
12 # print("URN: 2203398")
13 # print("Amanpreet kaur")
14
15 # num1 = int(input("Enter first number: "))
16 # op = input("Enter operator (+, -, *, /) ")
17 # num2 = int(input("Enter second number: "))
18 # if op == "+":
19 #     result = num1 + num2
20 # elif op == "-":
21 #     result = num1 - num2
22 # elif op == "*":
23 #     result = num1 * num2
24 # elif op == "/":
25 #     result = num1 / num2
26
27 print(result)

```

Output

```

URN: 2203398
Amanpreet kaur
The list is: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Odd index elements are:
1
3
5
7
9

```

## 1.12 Makeasimplecalculator. Makeasimplecalculator

Output:

The screenshot shows a web-based Python compiler interface. In the code editor (main.py), there is a Python script for a simple calculator. The output window shows the execution results: it prints the user's name, prompts for two numbers and an operator, performs the calculation, and handles division by zero. A message at the end indicates successful execution.

```
URN:2203398
Amanpreet kaur
Enter first number: 6
Enter operator (+, -, *, /): +
Enter second number: 6
6 + 6 = 12
== Code Execution Successful ===
```

## 1.13 Usingaloop to display elements from a given list present at odd index position.

```
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:Program to implement different data structure in Python.**

### **2.1 List**

Output:

The screenshot shows a Python script named `main.py` being run in an online compiler. The code performs various operations on a list, including printing, counting, indexing, sorting, and modifying elements. The output window displays the results of these operations.

```

20 print(my_list)
31 print("COUNT")
32 print(my_list.count(10)) #Count the repeated element in list
33 print("INDEX")
34 print(my_list.index(10)) #Finding the index of element 10
35 print("SORT")
36 my_list.sort() #Sorting the list in ascending order
37 print(my_list)
38 print("APPEND LEFT")
39
40 my_list.insert(0,0) #Adding element at the begining
41 print(my_list)
42 print("POP LEFT")
43
44 my_list.pop(0) #Removing element from the begining
45 print(my_list)
46 print("LENGTH")
47
48 print(len(my_list)) #Finding the length of list
49 print("CLEAR")
50 my_list.clear() #Clearing the list
51 print(my_list)
52
53
54
55
56
57

```

Output:

```

^ URN:2203398
Amanpreet kaur
List is: [1, 2, 3, 4, 5, 7, 6, 9, 8, 10]
Performing the following operations on List:
APPEND
[1, 2, 3, 4, 5, 7, 6, 9, 8, 10, 11]
POP
[1, 2, 3, 4, 5, 7, 6, 9, 8, 10]
INSERT
[1, 2, 3, 4, 5, 0, 7, 6, 9, 8, 10]
EXTEND
[1, 2, 3, 4, 5, 0, 7, 6, 9, 8, 10, 10, 15, 13, 12, 14, 16]
REMOVE
[1, 2, 3, 4, 5, 0, 7, 6, 9, 8, 10, 10, 15, 13, 12, 14]
DELETE
[1, 2, 3, 4, 5, 7, 6, 9, 8, 10, 10, 15, 13, 12, 14]
COUNT
2
INDEX
9
SORT
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15]
APPEND LEFT
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15]
POP LEFT
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15]
LENGTH
15
CLEAR

```

### 3 Practical no. 3Program to implement different data structure in Python.List:

Python Lists are just like the arrays, declared in other languages which are an ordered collection of data. It is very flexible as the items in list

```

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

my_list = [1,2,3,4,5,7,6,9,8,10]#Create list or using list()
print("List is:",my_list)
print("Performing the following operations on List:")
print("APPEND")
my_list.append(11) #Adding at the end
print(my_list)
print("POP")

```

```

my_list.pop() #Removing last element print(my_list)
print("INSERT")

my_list.insert(5,0) #Inserting 0 at index 5 print(my_list)
print("EXTEND")

my_list.extend([10,15,13,12,14,16]) #Adding list to the list print(my_list)
print("REMOVE")

my_list.remove(16) #Removing 16 from the list print(my_list)
print("DELETE")

del my_list[5] #Deleting element from index 5

print(my_list) print("COUNT")
print(my_list.count(10)) #Count the repeated element in list print("INDEX")
print(my_list.index(10)) #Finding the index of element 10 print("SORT")
my_list.sort() #Sorting the list in ascending order print(my_list)
print("APPEND LEFT")

my_list.insert(0,0) #Adding element at the beginning print(my_list)
print("POP LEFT")

my_list.pop(0) #Removing element from the beginning print(my_list)
print("LENGTH")

print(len(my_list)) #Finding the length of list print("CLEAR")
my_list.clear() #Clearing the list print(my_list)

```

### 3.1 Set:

Output:

```

4 my_set = {1,2,3,4,5,6,7,8,9,10}
5
6 my_set1 = {2,4,6,8,10,12,14,16,18,20}
7
8 print("Set is:",my_set)
9
10 print("Performing the following operation on set:")
11 print("ADD")
12 my_set.add(11) #Adding at the end
13 print(my_set)
14 print("DISCARD")
15
16 my_set.discard(5) #Removing the element
17 print(my_set)
18 print("REMOVE")
19
20 my_set.remove(7) #Removing the element
21 print(my_set)
22 print("UPDATE")
23
24 my_set.update([12,13,14,15]) #Adding another set in set
25 print(my_set)
26

```

Output:

```

URN:2203398
Amanpreet kaur
Set is: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Performing the following operation on set:
ADD
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
DISCARD
{1, 2, 3, 4, 6, 7, 8, 9, 10, 11}
REMOVE
{1, 2, 3, 4, 6, 8, 9, 10, 11}
UPDATE
{1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15}
UNION
{1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 20}
INTERSECTION
{2, 4, 6, 8, 10, 12, 14}
DIFFERENCE
{2, 4, 6, 8, 10, 12, 14}
SYMMETRIC DIFFERENCE
{1, 3, 9, 11, 13, 15, 16, 18, 20}
CLEAR
None

```

## 3.2 Set:

Python Set is an unordered collection of data that is mutable and does not allow any duplicate element. Sets are basically used to include membership testing and eliminating duplicate entries. The data structure used in this is Hashing, a popular technique to perform insertions, deletions, and lookups in O(1) time.

```

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("Set is:",my_set)

print("Performing the following operation on set:")print("ADD")
my_set.add(11) #Adding at the endprint(my_set)

```

```

print("DISCARD")

my_set.discard(5) #Removing the element print(my_set)
print("REMOVE")

my_set.remove(7) #Removing the element print(my_set)
print("UPDATE")

my_set.update([12,13,14,15]) #Adding another set in set print(my_set)
print("UNION")

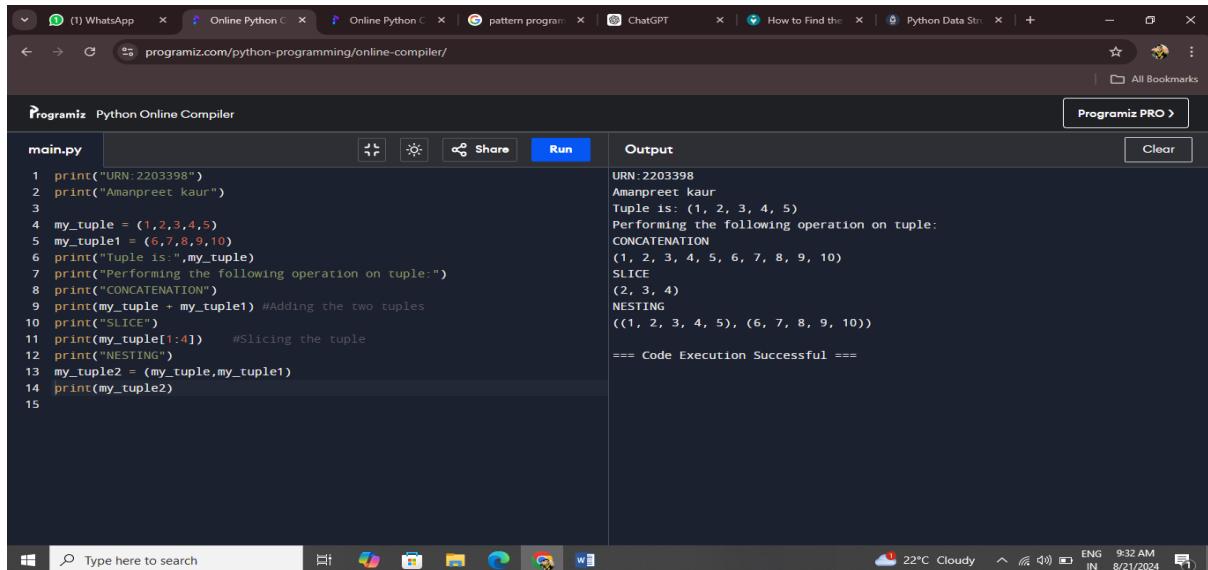
print(my_set.union(my_set1)) #Union of two sets print("INTERSECTION")
print(my_set.intersection(my_set1)) #Intersection of two sets print("DIFFERENCE")

print(my_set.intersection(my_set1)) #Intersection of two sets print("SYMMETRY DIFFERENCE")
print(my_set.symmetric_difference(my_set1)) #Different elements print("CLEAR")
print(my_set.clear()) #Clearing the set

```

### 3.3 Tuple

Output:



```

URN: 2203398
Amanpreet kaur
Tuple is: (1, 2, 3, 4, 5)
Performing the following operation on tuple:
CONCATENATION
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
SLICE
(2, 3, 4)
NESTING
((1, 2, 3, 4, 5), (6, 7, 8, 9, 10))
== Code Execution Successful ==

```

### **3.4 Tuple:**

Python Tuple is a collection of Python objects much like a list but Tuples are immutable in nature i.e. the elements in the tuple cannot be added or removed once created. Just like a List, a Tuple can also contain elements of various types.

```
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:

```

main.py
1 print("URN:2203398")
2 print("Amanpreet kaur")
3
4 print("URN:2203395")
5 my_dict = {1:"Aman",2:"Gur",3:"Kirat"}
6 print("Performing the following operations on the dictionary:")
7 print("GET")
8 print(my_dict.get(2)) #Return the value of the key
9 print("ITEMS")
10 print(my_dict.items()) #Return the key value paired
11 print("KEYS")
12 print(my_dict.keys()) #Return the keys of the dictionary
13 print("VALUES")
14 print(my_dict.values()) #Return the values of the dictionary
15 print("POP")
16 print(my_dict.pop(3)) #Remove the key value pair
17 print(my_dict)
18 print("UPDATE")
19 my_dict.update({4:"Aashima"}) #Add the key value pair
20 print(my_dict)
21 print("CLEAR")
22 my_dict.clear() #Remove all the key value pairs

```

Output

```

URN 2203398
Amanpreet kaur
URN 2203395
Performing the following operations on the dictionary:
GET
Gur
ITEMS
dict_items([(1, 'Aman'), (2, 'Gur'), (3, 'Kirat')])
KEYS
dict_keys([1, 2, 3])
VALUES
dict_values(['Aman', 'Gur', 'Kirat'])
POP
Kirat
{1: 'Aman', 2: 'Gur'}
UPDATE
{1: 'Aman', 2: 'Gur', 4: 'Aashima'}
CLEAR
{}

== Code Execution Successful ==

```

## Dictionary:

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-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.