

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

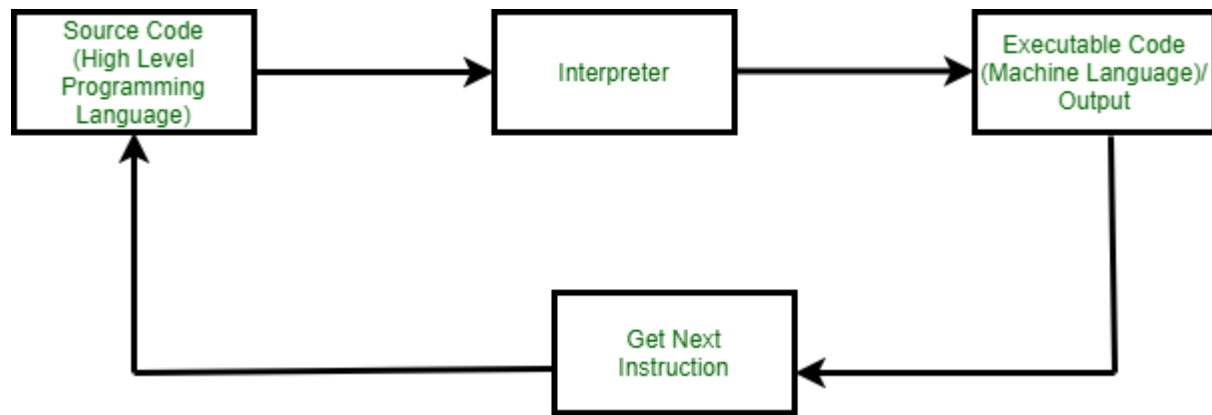
0.2 Features of python:

- Free and open source
- Easy to code
- Easy to read
- Object oriented language
- GUI programming support
- High-level language
- Large community support
- Easy to debug
- Python is a portable language
- Python is an integrated language
- Dynamically typed language

0.3 Difference between compiler and interpreter:

S.NO.	Compiler	Interpreter
1.	<p>Steps of Programming:</p> <ul style="list-style-type: none"> • Program Creation. • Analysis of language by the compiler and throw errors in case of any file or statement. • In case of no error, the Compiler converts the source code to Machine Code. • Linking of various code files into a runnable program. • Finally runs a Program. 	<p>Steps of Programming:</p> <ul style="list-style-type: none"> • Program Creation. • Machine Code is not required. • Machine Code source statements one by one.
2.	The compiler saves the Machine Language in the form of Machine Code on the hard disk.	The Interpreter does not save the Machine Language.
3.	Compiled codes run faster than Interpreter.	Interpreted codes run slower than Compiler.
4.	Linking-Loading Model is the basic working model of the Compiler.	One Interpretation Model is the basic working model of the Interpreter.
5.	The compiler 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 entire code.
7.	Errors are displayed in the Compiler after compiling together.	Errors 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 execute each line of statements slowly. This process is called Interpretation. For example Python is an interpreted language, PHP, Ruby, and Java Script.



Working of Interpreter

0.4 Python versions:

1. Python 3.12.4, documentation released on 6 June 2024.
2. Python 3.12.3, documentation released on 9 April 2024.
3. Python 3.12.2, documentation released on 6 February 2024.
4. Python 3.12.1, documentation released on 8 December 2023.
5. Python 3.12.0, documentation released on 2 October 2023.
6. Python 3.11.9, documentation released on 2 April 2024.
7. Python 3.11.8, documentation released on 6 February 2024.
8. Python 3.11.7, documentation released on 4 December 2023.
9. Python 3.11.6, documentation released on 2 October 2023.
10. Python 3.11.5, documentation released on 24 August 2023.

0.5 Python IDE:

- PyCharm
- IDLE
- Visual Studio Code
- Atom
- Sublime Text

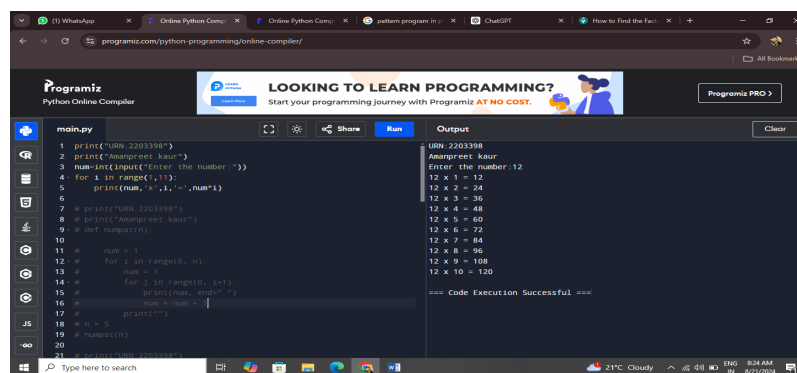
- Spyder
- PyDev

1 Practicalno

Programstoimplement inputoutputandcontrolflowtoolsinpython.

1.1 Multiplicationtableofthenumberenteredbytheuser

Output:



The screenshot shows the Programiz Python Online Compiler interface. The code in the editor is as follows:

```

1 print("URN 2203398")
2 print("Amangreet 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("URN 2203398")
8 # print("Amangreet kaur")
9 # def numpat(n):
10
11 #     num = 1
12 #     for i in range(0, n):
13 #         num = i + 1
14 #         for j in range(0, i+1):
15 #             print(num, end=" ")
16 #             num = num + 1
17 #         print("\n")
18 #     n = 5
19 #     numpat(n)
20
21 # def main():

```

The output on the right shows the execution results:

```

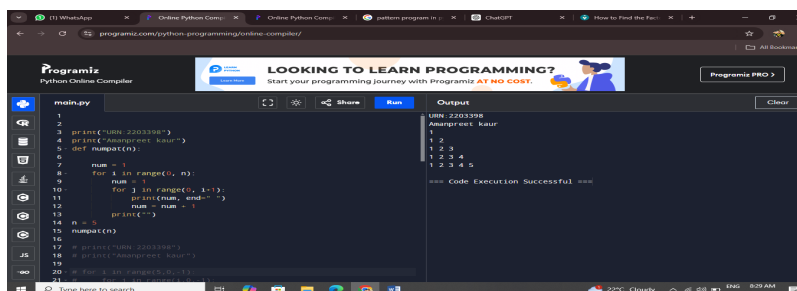
URN 2203398
Amangreet kaur
Enter the number: 12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132

=== Code Execution Successful ===

```

1.2 Printthepattern:

Output:



The screenshot shows the Programiz Python Online Compiler interface. The code in the editor is as follows:

```

1
2 print("URN 2203398")
3 print("Amangreet kaur")
4 def numpat(n):
5
6     num = 1
7     for i in range(0, n):
8         num = i + 1
9         for j in range(0, i+1):
10             print(num, end=" ")
11             num = num + 1
12         print("\n")
13
14 n = 5
15 numpat(n)
16
17 # print("URN 2203398")
18 # print("Amangreet kaur")
19
20 # def main():
21

```

The output on the right shows the execution results:

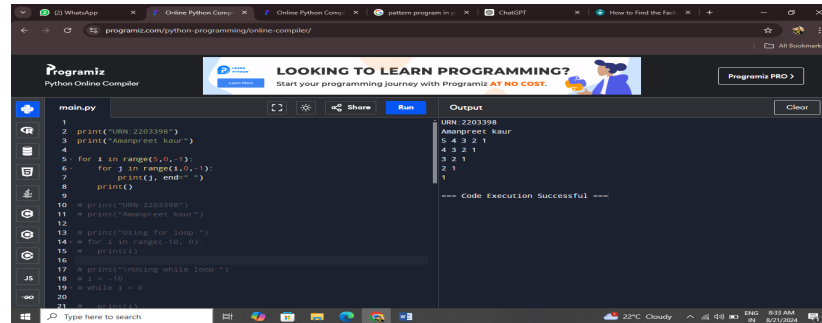
```

URN 2203398
Amangreet kaur
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

=== Code Execution Successful ===

```

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:

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

```

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 editor on the left contains a Python script (main.py) with the following content:

```

1
2 print("URN:2203398")
3 print("Amanpreet kaur")
4
5 print("Using for loop:")
6 for i in range(-10, 0):
7     print(i)
8
9 print("\nUsing while loop:")
10 i = -10
11 while i < 0:
12     print(i)
13     i += 1
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)

```

The output panel on the right shows the following 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 the Programiz Python Online Compiler interface. The code editor on the left contains a Python script (main.py) with the following content:

```

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 # print("URN:2203398")
21 # print("Amanpreet kaur")
22

```

The output panel on the right shows the following 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("\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

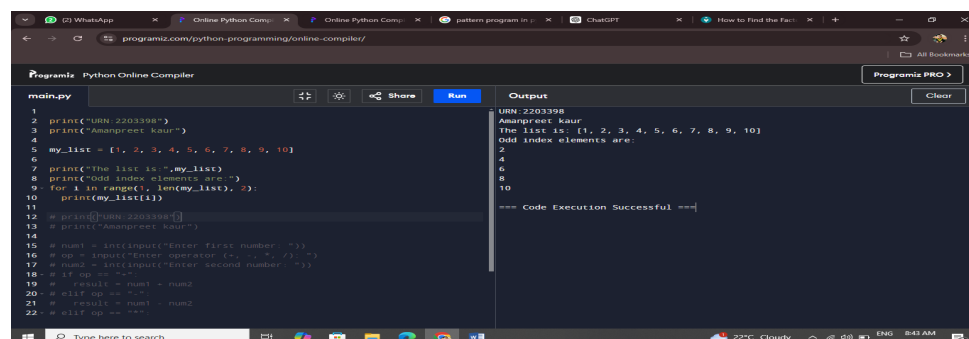
```
print("URN:2203398")
print("Amanpreet kaur")
```

```
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:

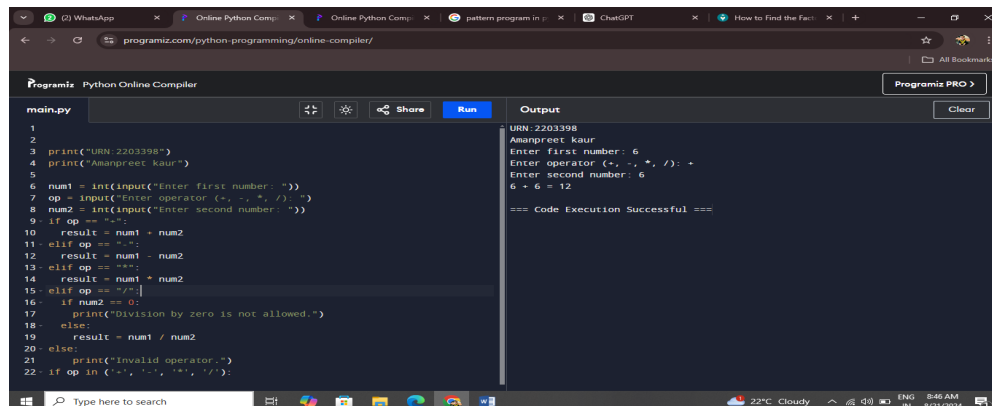


```
main.py
1
2 print("URN:2203398")
3 print("Amanpreet kaur")
4
5 my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
6
7 print("The list is:",my_list)
8 print("Odd index elements are:")
9 for i in range(1, len(my_list), 2):
10     print(my_list[i])
11
12 # print(URN:2203398)
13 # print(Amanpreet kaur)
14
15 # 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 == "*":
```

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

1.12 Makeasimplecalculator. Makeasimplecalculator

Output:



The screenshot shows a web browser with the URL `programiz.com/python-programming/online-compiler/`. The page displays a Python code editor with a file named `main.py`. The code is a simple calculator that takes two numbers and an operator as input and performs the corresponding arithmetic operation. The output window on the right shows the execution results, including the program's output and the user's input.

```
main.py
1
2
3 print("URN:2203398")
4 print("Amanpreet kaur")
5
6 num1 = int(input("Enter first number: "))
7 op = input("Enter operator (+, -, *, /): ")
8 num2 = int(input("Enter second number: "))
9
10 if op == "+":
11     result = num1 + num2
12 elif op == "-":
13     result = num1 - num2
14 elif op == "*":
15     result = num1 * num2
16 elif op == "/":
17     if num2 == 0:
18         print("Division by zero is not allowed.")
19     else:
20         result = num1 / num2
21 else:
22     print("Invalid operator.")
23 if op in ("+", "-", "*", "/"):
24     print(f"{num1} {op} {num2} = {result}")
25 else:
26     print("Invalid operator.")
```

Output

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

=== Code Execution Successful ===
```

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:ProgramtoimplementdifferentdatastructureinPython.

2.1 List

Output:

Programiz Python Online Compiler
Programiz PRO >

main.py

Share

Run

Output

Clear

```

30 print(my_list)
31 print("COUNT")
32 print(my_list.count(10)) #Count the repeted 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 begning
41 print(my_list)
42 print("POP LEFT")
43
44 my_list.pop(0) #Removing element from the begning
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

```

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. 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("PerformingthefollowingoperationsonList:")
print("APPEND")
my_list.append(11) #Addingattheend
print(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 thebeginningprint(my_list)
print("POP LEFT")

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

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

```

3.1 Set:

Output:

The screenshot shows the Programiz Python Online Compiler interface. The left pane contains the following Python code:

```

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("Perfroming 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

```

The right pane shows the output of the script:

```

URN:2203398
Amanpreet kaur
Set is: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Perfroming 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}
SYMMETRY 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 and deletions in O(1) time complexity.

```
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("Perfroming the following operation on set:")
print("ADD")
my_set.add(11) #Adding at the end
print(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]) #Addinganother setinsetprint(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:

```

main.py
1 print("URN: 2203398")
2 print("Amanpreet kaur")
3
4 my_tuple = (1,2,3,4,5)
5 my_tuple1 = (6,7,8,9,10)
6 print("Tuple is:",my_tuple)
7 print("Performing the following operation on tuple:")
8 print("CONCATENATION")
9 print(my_tuple + my_tuple1) #Adding the two tuples
10 print("SLICE")
11 print(my_tuple[1:4]) #slicing the tuple
12 print("NESTING")
13 my_tuple2 = (my_tuple,my_tuple1)
14 print(my_tuple2)
15
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:

The screenshot shows the Programiz Python Online Compiler interface. The left pane contains a Python script named `main.py` with 22 lines of code. The right pane shows the output of the script, which includes the execution of various dictionary methods and their results. The output is as follows:

```
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.