

Python Assignment-1

1) Writing and Running First Program

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists several Python files. In the center, the code editor displays a file named 'First program.py' with the following content:

```
# Creating and First program
print("Welcome to python")
```

Below the code editor, the 'Run' tool window shows the command: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/First program.py". The output pane shows the result: Welcome to python. Process finished with exit code 0.

2) Key words

Keywords in Python are reserved words that can not be used as a variable name, function name, or any other identifier.

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists several Python files. In the center, the code editor displays a file named 'keywords.py' with the following content:

```
# Keywords
# Print all the keywords in python
import keyword
print(keyword.kwlist)
print("The keywords in python are:")
count=0
for i in keyword.kwlist:
    print(i)
    count=count+1
print(f"Total number of keywords:{count}")

for i in keyword.kwlist
```

Below the code editor, the 'Run' tool window shows the command: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/keywords.py". The output pane shows the result: The keywords in python are: ['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from']

3) Identifiers

- **Identifier** is a user-defined name given to a variable, function, class, module, etc.
- The identifier is a combination of character digits and an underscore. They are case-sensitive.

```
# Identifiers
# Rules for identifiers:
# 1)Identifiers can be combination of uppercase and lowercase letters, digits or an underscore(_).
# 2)An Identifier can not start with digit. So while variable1 is valid, 1variable is not valid.
# 3)We can't use special symbols like !,@,#,$ etc in our Identifier.
# 4)Identifier can be of any length.

# Valid Identifier
_xyz=1111
print(_xyz)

set_Value=89.098
print(set_Value)

#Identifier can be of any length
hhhhhhhhhhhhhh222222222222="Hello"
print(hhhhhhhhhhhhh222222222222)

#Invalid Identifier
#@sum=111
#@print(@sum)
```

The screenshot shows the PyCharm IDE interface with the 'Identifiers.py' file open. The code examples illustrate various identifier rules. The output window shows the results of running the script: 1111, 89.098, Hello, and an error message for the invalid identifier example.

4) Variables

- Python Variable is containers that store values.
- Python is not “statically typed”.
- We do not need to declare variables before using them or declare their type.

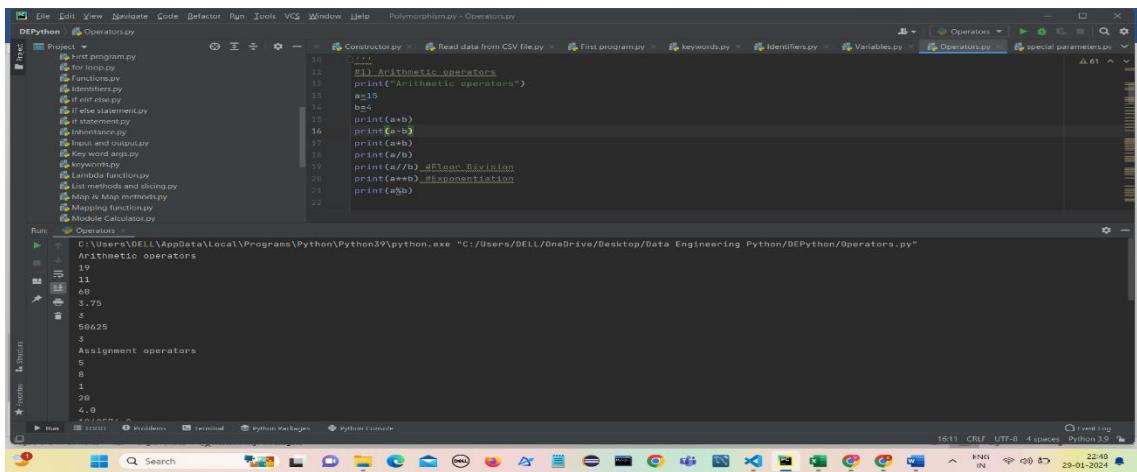
```
#Variables
print("Variables")
var="Hello"
print(var)

value=345.90
print(value)
```

The screenshot shows the PyCharm IDE interface with the 'Variables.py' file open. The code defines a string variable 'var' and a float variable 'value', then prints them. The output window shows the results: Hello and 345.9.

5) Operators

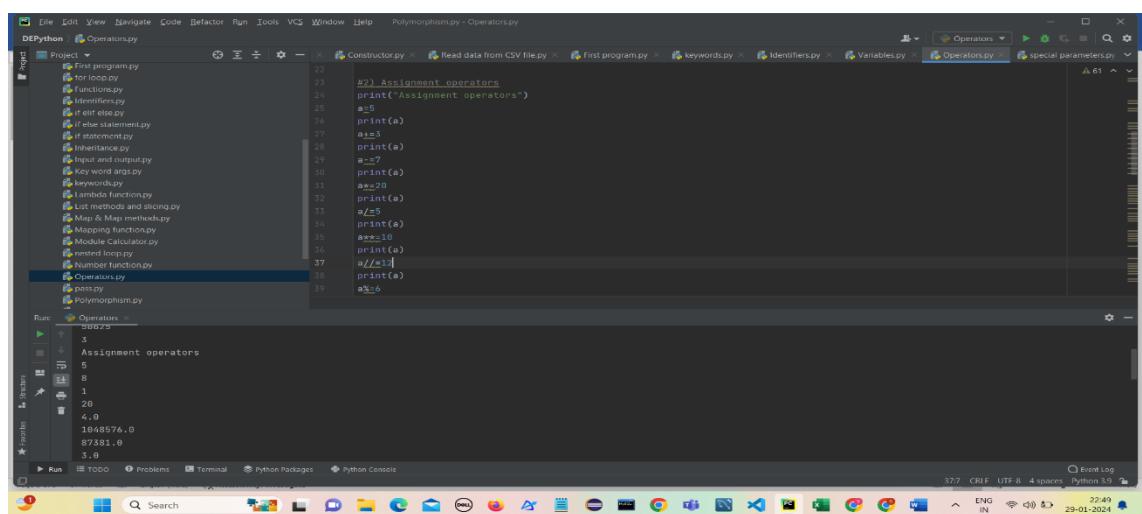
i) Arithmetic Operators



```
DEPython Operators.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - Operators.py
Project Operators
  First program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  if else statement.py
  if statement.py
  Inheritance.py
  Input and output.py
  Key word args.py
  Lambda functions.py
  List methods and slicing.py
  Map & Map methods.py
  Nested loop.py
  Module Calculator.py
  Operators.py
  Polymorphism.py
Run Operators
C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Operators.py"
3
11
a0
3.75
3
50625
3
Assignment operators
5
8
1
20
4.0
Run 20023
16
print(a+b)
17
print(a-b)
18
print(a*b)
19
print(a/b) #Error: Division
20
print(a**b) #Exponentiation
21
print(a%b)
22
```

The screenshot shows the PyCharm IDE interface with the code for arithmetic operators. The code includes basic operations like addition, subtraction, multiplication, division, exponentiation, and modulus. A comment in the code indicates that division results in an error. The run output shows the expected numerical results for each operation.

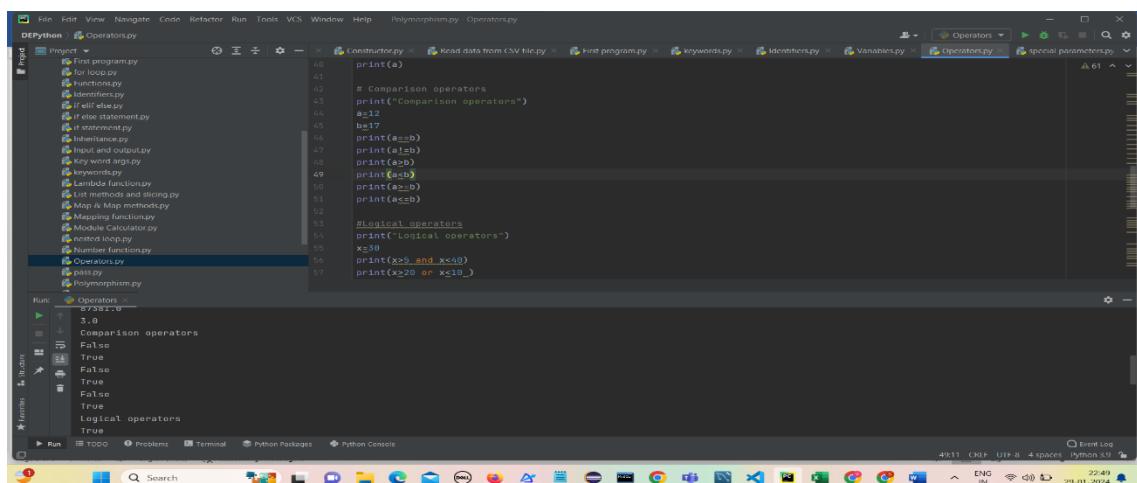
ii) Assignment Operators



```
DEPython Operators.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - Operators.py
Project Operators
  First program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  if else statement.py
  if statement.py
  Inheritance.py
  Input and output.py
  Key word args.py
  Lambda functions.py
  List methods and slicing.py
  Map & Map methods.py
  Nested loop.py
  Number function.py
  Operators.py
  Polymorphism.py
Run Operators
20023
23
#2) Assignment operators
24
print("Assignment operators")
25
a=5
26
print(a)
27
a+=3
28
print(a)
29
a-=7
30
print(a)
31
a*=20
32
print(a)
33
a/=5
34
print(a)
35
a**=10
36
print(a)
37
a//=1
38
print(a)
39
a%=5
```

The screenshot shows the PyCharm IDE interface with the code for assignment operators. The code demonstrates various assignment operators such as +=, -=, *=, /=, **=, //=, and %=. The run output shows the final value of 'a' after each assignment operation.

iii) Comparison Operators

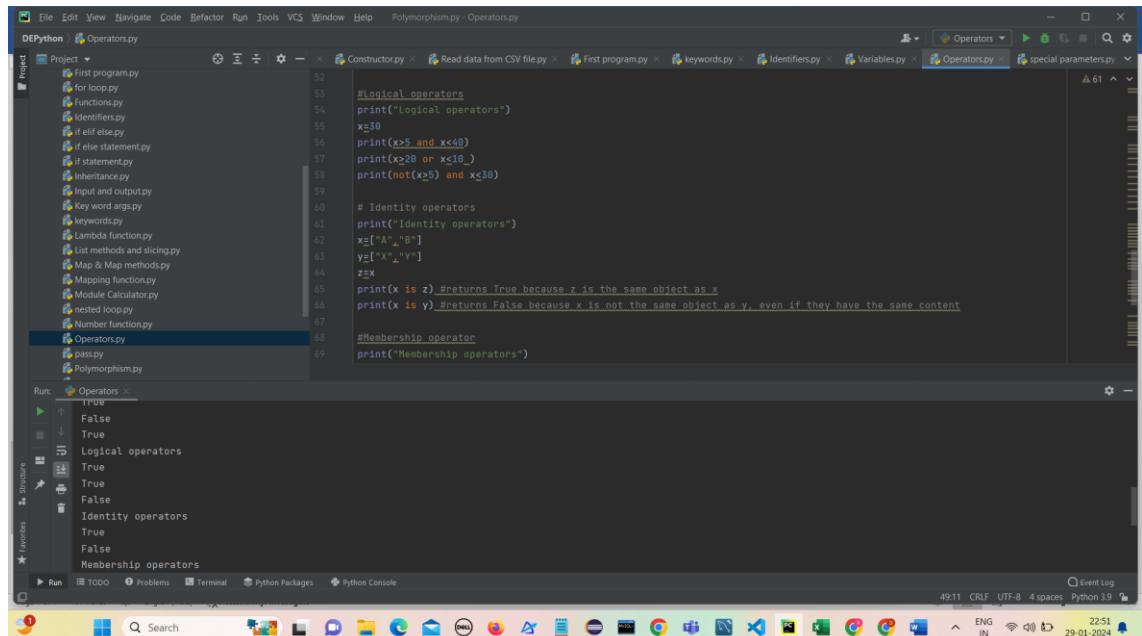


```
DEPython Operators.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - Operators.py
Project Operators
  First program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  if else statement.py
  if statement.py
  Inheritance.py
  Input and output.py
  Key word args.py
  Lambda functions.py
  List methods and slicing.py
  Map & Map methods.py
  Nested loop.py
  Number function.py
  Operators.py
  Polymorphism.py
Run Operators
20023
3.0
Comparison operators
False
True
False
True
Logical operators
True
Run 20023
40
print(a)
41
# Comparison operators
42
print("Comparison operators")
43
a=12
44
b=17
45
print(a==b)
46
print(a!=b)
47
print(a>b)
48
print(a<b)
49
print(a>=b)
50
print(a<=b)
51
Logical operators
52
print("Logical operators")
53
x=9
54
print(x>5 and x<10)
55
print(x>20 or x<10)
```

The screenshot shows the PyCharm IDE interface with the code for comparison and logical operators. The code uses operators like ==, !=, >, <, >=, <=, and logical operators like and and or. The run output shows the boolean results of these comparisons.

iv) Logical Operators

v) Identity Operators



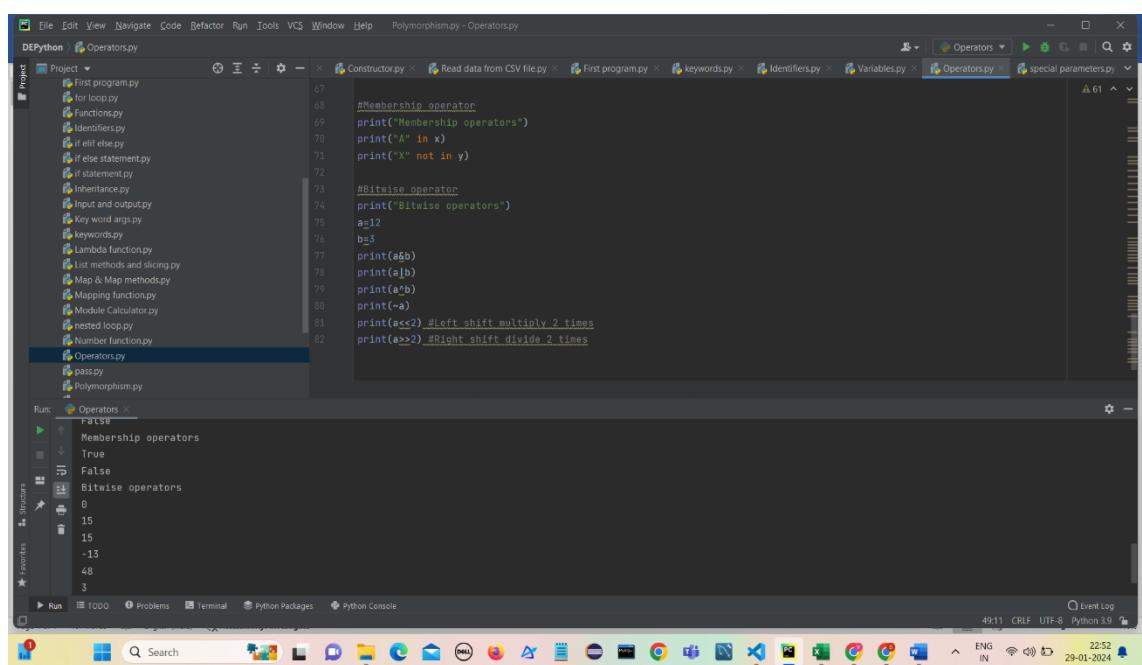
The screenshot shows the PyCharm IDE interface with the project 'Operators.py' open. The code editor displays Python code demonstrating logical operators:

```
52 #Logical operators
53 print("Logical operators")
54 x=50
55 print(x>5 and x<40)
56 print(x>20 or x<10)
57 print(not(x>5) and x<30)
58
59 # Identity operators
60 print("Identity operators")
61 x=["A","B"]
62 y=[x,"Y"]
63 z=x
64 print(x is z) #returns True because z is the same object as x
65 print(x is y) #returns False because x is not the same object as y, even if they have the same content
66
67 #Membership operator
68 print("Membership operators")
```

The code editor has a sidebar with a tree view of the project structure, showing files like First program.py, for loop.py, Functions.py, Identifiers.py, etc. The 'Run' tab is selected, showing the output of the code execution.

vi) Membership Operators

vii) Bitwise Operators



The screenshot shows the PyCharm IDE interface with the project 'Operators.py' open. The code editor displays Python code demonstrating bitwise operators:

```
67 #Membership operator
68 print("Membership operators")
69 print("A" in x)
70 print("X" not in y)
71
72 #Bitwise operator
73 print(a&b)
74 print(a|b)
75 print(a^b)
76 print(~a)
77 print(a<>2) #Left shift multiply 2 times
78 print(a>>2) #Right shift divide 2 times
79
80
81
82
```

The code editor has a sidebar with a tree view of the project structure, showing files like First program.py, for loop.py, Functions.py, Identifiers.py, etc. The 'Run' tab is selected, showing the output of the code execution.

6) Datatypes

```
DEPython > Datatypes.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - Datatypes.py
Project > Datatypes.py
  Constructors.py
  continue.py
  Datatypes.py
  Date and time function.py
  Default arguments.py
  Dictionary and dict methods.py
  Example_1.csv
  extra.py
  first program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  Input and output.py
  Key word args.py
  keywords.py
  Lambda functions.py
  List methods and dicrionary
Run: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Datatypes.py"
Datatypes
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'list'>
<class 'tuple'>
<class 'set'>
<class 'dict'>
<class 'str'>
<class 'bool'>
print(type(12))
print(type(12.99))
print(type(1+2j))
print(type([1,2,3]))
print(type((1,2,3)))
print(type({1,2,3,4,5}))
print(type({'name': 'apple'}))
print(type('apple'))
#Tuple
t=(12,99,34,55)
print(type(t))
#Set
s={1,2,3,4,5}
print(type(s))
#Dictionary
d={'name': 'apple'}
print(type(d))

Process finished with exit code 0
22:54 29-01-2024
```

Control Structures

7) If statement

```
DEPython > if statement.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - if statement.py
Project > if statement.py
  Constructors.py
  continue.py
  Datatypes.py
  Date and time function.py
  Default arguments.py
  Dictionary and dict methods.py
  Example_1.csv
  extra.py
  first program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  if statement.py
  Inheritance.py
  Input and output.py
  Key word args.py
  keywords.py
  Lambda functions.py
  List methods and dicrionary
Run: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/if statement.py"
Enter age:18
Eligible to vote
Process finished with exit code 0
22:55 29-01-2024
```

8) If else statement

```
DEPython > if else statement.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - if else statement.py
Project > if else statement.py
  Constructors.py
  continue.py
  Datatypes.py
  Date and time function.py
  Default arguments.py
  Dictionary and dict methods.py
  Example_1.csv
  extra.py
  first program.py
  for loop.py
  Functions.py
  Identifiers.py
  if elif else.py
  if else statement.py
  Inheritance.py
  Input and output.py
  Key word args.py
  keywords.py
  Lambda functions.py
  List methods and dicrionary
Run: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/if else statement.py"
Enter a number:4
4 is even number
Process finished with exit code 0
22:55 29-01-2024
```

9) If elif else statement

The screenshot shows the PyCharm IDE interface with the code editor open. The file name is 'if elif else.py'. The code defines a function 'first program' that takes marks as input and prints the grade based on the marks. The IDE's status bar at the bottom right shows the date as 29-01-2024 and the time as 22:57.

```
DEPython if elif else.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - if elif else.py
Project
  Constructors.py
  continue.py
  Datatypes.py
  Date and time functions.py
  Dictionary and dict methods.py
  Example_1.csv
  exit.py
  exception.py
  for loop.py
  Functions.py
  Identifiers.py
  If elif else.py
  If statement.py
  Inheritance.py
  Input and output.py
  Lambda functions.py
  keywords.py
  Lambda functions.py
  List methods and slicing.py
  Map & Map methods.py
  Mapping functions.py
  Module Calculator.py
  nested loop.py
  Number functions.py
  Operators.py
  pass.py
  Polymorphism.py
  Read data from CSV file.py
  Set & Set methods.py
  special parameters.py
  String Functions.py
  tuple.py
  Variables.py
  while loop.py
Scratches and Consoles
Run: while loop
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/if elif else.py"
Enter marks: 90
B grade
Process finished with exit code 0
Run: Run TODO Problems Terminal Python Packages Python Console Event Log 10:20 Python 3.9
File Run TODO Problems Terminal Python Packages Python Console Event Log 22:57 29-01-2024
```

```
1
2 # If elif else statement
3 marks=int(input("Enter marks:"))
4
5 if marks>=75:
6     print("A grade")
7 elif marks>=65 and marks<75:
8     print("B grade")
9 elif marks>=55 and marks<65:
10    print("C grade")
11 else:
12     print("D grade")
```

10) For Loop

The screenshot shows the PyCharm IDE interface with the code editor open. The file name is 'for loop.py'. The code uses a for loop to print even numbers from 1 to 100. The IDE's status bar at the bottom right shows the date as 29-01-2024 and the time as 22:01.

```
DEPython for loop.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - for loop.py
Project
  Constructors.py
  continue.py
  Datatypes.py
  Date and time functions.py
  Dictionary and dict methods.py
  Example_1.csv
  exit.py
  exception.py
  for loop.py
  Functions.py
  Identifiers.py
  If elif else.py
  If statement.py
  Inheritance.py
  Input and output.py
  Lambda functions.py
  keywords.py
  Lambda functions.py
  List methods and slicing.py
  Map & Map methods.py
  Mapping functions.py
  Module Calculator.py
  nested loop.py
  Number functions.py
  Operators.py
  pass.py
  Polymorphism.py
  Read data from CSV file.py
  Set & Set methods.py
  special parameters.py
  String Functions.py
  tuple.py
  Variables.py
  while loop.py
Scratches and Consoles
Run: for loop
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/for loop.py"
Even numbers from 1 to 100:
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98
1 3 5 7 9 11 13 15 17 19 21 23 25
Process finished with exit code 0
Run: Run TODO Problems Terminal Python Packages Python Console Event Log 22:01 Python 3.9
File Run TODO Problems Terminal Python Packages Python Console Event Log 22:01 29-01-2024
```

```
1
2 # for loop
3
4 for i in range(1,100):
5     if i%2==0:
6         print(i,end=' ')
7     else:
8         print(i,end=' ')
9
10 for i in range(1,26,2):
11     print(i,end=' ')
```

11) While loop

The screenshot shows the PyCharm IDE interface with the code editor open. The file name is 'while loop.py'. The code uses a while loop to print numbers from 1 to n. The IDE's status bar at the bottom right shows the date as 29-01-2024 and the time as 23:02.

```
DEPython while loop.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - while loop.py
Project
  Key word args.py
  keywords.py
  Lambda function.py
  List methods and slicing.py
  Map & Map methods.py
  Mapping functions.py
  Module Calculator.py
  nested loop.py
  Number functions.py
  Operators.py
  pass.py
  Polymorphism.py
  Read data from CSV file.py
  Set & Set methods.py
  special parameters.py
  String Functions.py
  tuple.py
  Variables.py
  while loop.py
Scratches and Consoles
Run: while loop
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/while loop.py"
Enter n:25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Process finished with exit code 0
Run: Run TODO Problems Terminal Python Packages Python Console Event Log 23:02 Python 3.9
File Run TODO Problems Terminal Python Packages Python Console Event Log 23:02 29-01-2024
```

```
1
2 #while loop
3 n=int(input("Enter n:"))
4 i=1
5 while(i<n):
6     print(i,end=' ')
7     i+=1
```

12) Nested loop

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists various Python files. The main editor window contains the following code:

```
1 #Nested loop
2     n=int(input("Enter a number:"))
3     for i in range(1,n):
4         for j in range(1,i):
5             print(i,j,end=" ")
6
7 #Multiplication table using nested loops
8 print('\n')
9 print("Multiplication tables from 1 to 5:")
10 for i in range(1,6):
11     for j in range(1,11):
12         print(f'{i}{j}={i*j}')
13     print()
14
15 for i in range(1,6):
```

The 'Run' tab in the bottom left shows the command: C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/nested loop.py". The terminal output shows the multiplication tables from 1 to 5:

```
Enter a number:5
1 1 2 2 3 3 4 4 5 5
Multiplication tables from 1 to 5:
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
1*10=10
```

13) Break

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists various Python files. The main editor window contains the following code:

```
1 #break
2     print("Break")
3     for i in range(1,10):
4         if i==4:
5             break
6         print(f'{i} and')
```

The 'Run' tab in the bottom left shows the command: C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/break.py". The terminal output shows the numbers 1 through 3:

```
Break
1 2 3
Process finished with exit code 0
```

14) Continue

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists various Python files. The main editor window contains the following code:

```
1 #Continue
2     print("Continue")
3     for i in range(1,10):
4         if i==6:
5             continue
6         print(f'{i} and')
```

The 'Run' tab in the bottom left shows the command: C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/continue.py". The terminal output shows the numbers 1 through 9, skipping the number 6:

```
Continue
1 2 3 4 5 7 8 9
Process finished with exit code 0
```

15) Pass

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists several files: Input and output.py, Keyword args.py, keywords.py, Lambda function.py, List methods and slicing.py, Map & Map methods.py, Mapping function.py, Module Calculator.py, nested loop.py, Number function.py, Operators.py, pass.py, Polymorphism.py, Read data from CSV file.py, Set & Set methods.py, and special parameters.py. Below the navigation bar, the code editor displays the 'pass.py' file:

```
1 # pass
2 print("Pass")
3 print("Odd numbers")
4 n=int(input("Enter n:"))
5 for i in range(n):
6     if i%2==0:
7         pass
8     else:
9         print(i,end=' ')
for i in range(n)
```

The run tab below the code editor shows the command: C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/pass.py". The terminal window shows the output of the program:

```
Pass
Odd numbers
Enter n:57
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55
Process finished with exit code 0
```

The status bar at the bottom right indicates the time as 7:1 and the Python version as Python 3.9.

16) Input and Output

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists several files: Functions.py, Identifiers.py, If elif else.py, If else statement.py, If statement.py, Inheritance.py, Input and output.py, Key word args.py, keywords.py, Lambda function.py, List methods and slicing.py, Map & Map methods.py, Mapping function.py, Module Calculator.py, nested loop.py, Number function.py. Below the navigation bar, the code editor displays the 'Input and output.py' file:

```
1 #input and output
2 #single input
3 n=int(input("Enter n:"))
4 print(f'Value of n:{n}')
5
6 #multiple input
7 # map-maps the value with int
8 list=list(map(int,input("Enter values").split()))
9
10 print(list)
11
12
```

The run tab below the code editor shows the command: C:/Users/DELL/AppData/Local/Programs/Python/Python39/python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Input and output.py". The terminal window shows the output of the program:

```
Enter n:789
Value of n:789
Enter values12 45 499 2647 100000
[12, 45, 499, 2647, 100000]
Process finished with exit code 0
```

The status bar at the bottom right indicates the time as 8:1 and the Python version as Python 3.9.

17) List methods and Slicing

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - List methods and slicing.py. The left sidebar has a Project view with various Python files listed. The main code editor window contains the following Python code:

```
10     """
11     print("List")
12     l1=[10,20,30,40,50,10.5,10,20]
13     l2=[100,200,300]
14     l1.extend(l2)
15     print(dir(l1))
16     print(l1.count(10))
17     print(l1.index(40))
18     #function of list:
19     l1.append(45)#add item in last
20     print(l1)
21     l1.remove(10.5)
22     print(l1)
23     l1.pop()#delete last element
24     print(l1)
25     l1.insert(2,99)
26     print(l1)
27     l1.reverse()#reverse the list
```

The Run tab shows the command: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/List methods and slicing.py". The output window displays the results of the list operations.

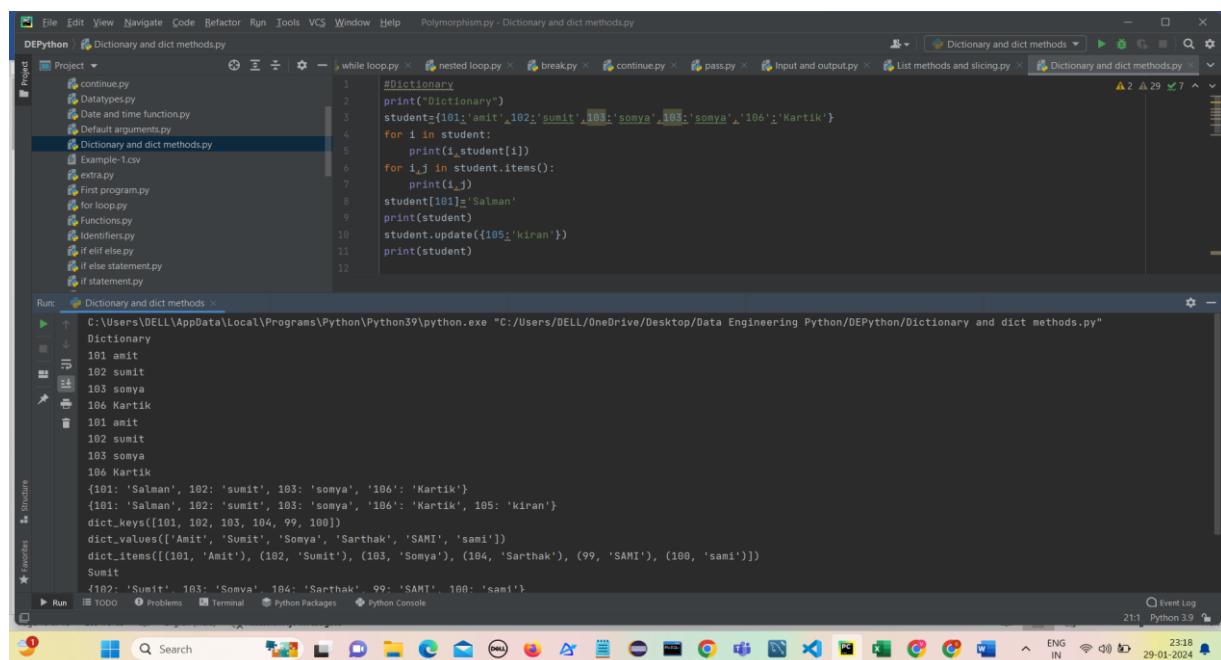
The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - List methods and slicing.py. The left sidebar has a Project view with various Python files listed. The main code editor window contains the following Python code:

```
30     print(l1)
31     l1.extend(l2)
32     print(l1) #extend
33
34     #slicing
35     print("Slicing in list")
36     l2=[11,34.5,"Welcome",12,1,4,7,9,111,2222,989899]
37     print(l2)
38     print(l2[2:7])
39     print(l2[::-1]) # whole list
40     print(l2[::-1]) #reverse
41     print(l2[-5:-1])
```

The Run tab shows the command: C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/List methods and slicing.py". The output window displays the results of the list slicing operations.

18) Dictionary and Dictionary methods

```
    #!/usr/bin/python  
  
d.keys()#return the all the keys  
d.values()#return the all values  
d.items()#return the key and value in pair  
d.clear()#delete all key and values  
d.get(101)#return the values  
d.update({106:'Likitha'})#add the pair in the dict  
d.pop(99)#delete the given key from the dict  
d.popitem()#delete the last pair from dict  
    
```

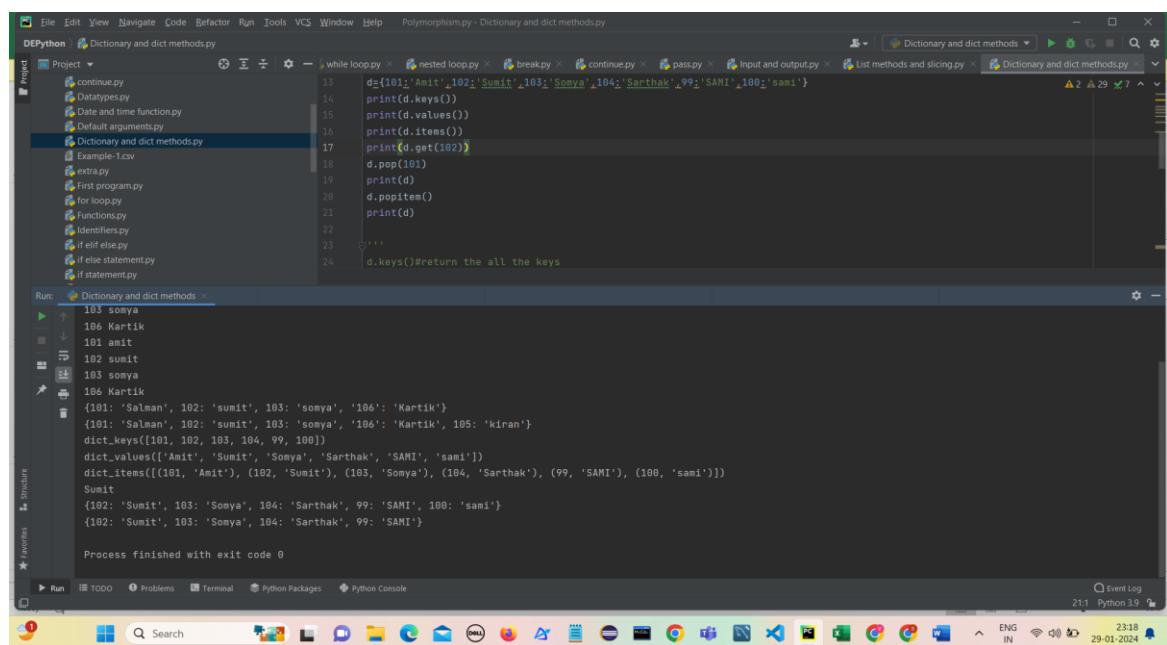


The screenshot shows a Python IDE interface with a code editor and a run terminal. The code in the editor is:

```
#Dictionary  
student={101:'amit', 102:'sumit', 103:'somya', 104:'Somya', 105:'Kartik'}  
for i in student:  
    print(i,student[i])  
for i,j in student.items():  
    print(i,j)  
student[101]='Salman'  
print(student)  
student.update({105:'kiran'})  
print(student)
```

The run terminal shows the output of the code:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Dictionary and dict methods.py"  
Dictionary  
101 amit  
102 sumit  
103 somya  
106 Kartik  
101 amit  
102 sumit  
103 somya  
106 Kartik  
{101: 'Salman', 102: 'sumit', 103: 'somya', 106: 'Kartik'}  
{101: 'Salman', 102: 'sumit', 103: 'somya', 106: 'Kartik', 105: 'kiran'}  
dict_keys([101, 102, 103, 104, 99, 100])  
dict_values(['Amit', 'Sumit', 'Somya', 'Sarthak', 'SAMI', 'sami'])  
dict_items([(101, 'Amit'), (102, 'Sumit'), (103, 'Somya'), (104, 'Sarthak'), (99, 'SAMI'), (100, 'sami')])  
Sumit  
{102: 'Sumit', 103: 'Somya', 104: 'Sarthak', 99: 'SAMI', 100: 'sami'}  
Process finished with exit code 0
```



The screenshot shows a Python IDE interface with a code editor and a run terminal. The code in the editor is:

```
d={101:'Amit',102:'Sumit',103:'Somya',104:'Sarthak',99:'SAMI',100:'sami'}  
print(d.keys())  
print(d.values())  
print(d.items())  
print(d.get(102))  
d.pop(101)  
print(d)  
d.popitem()  
print(d)  
    #d.keys()#return the all the keys
```

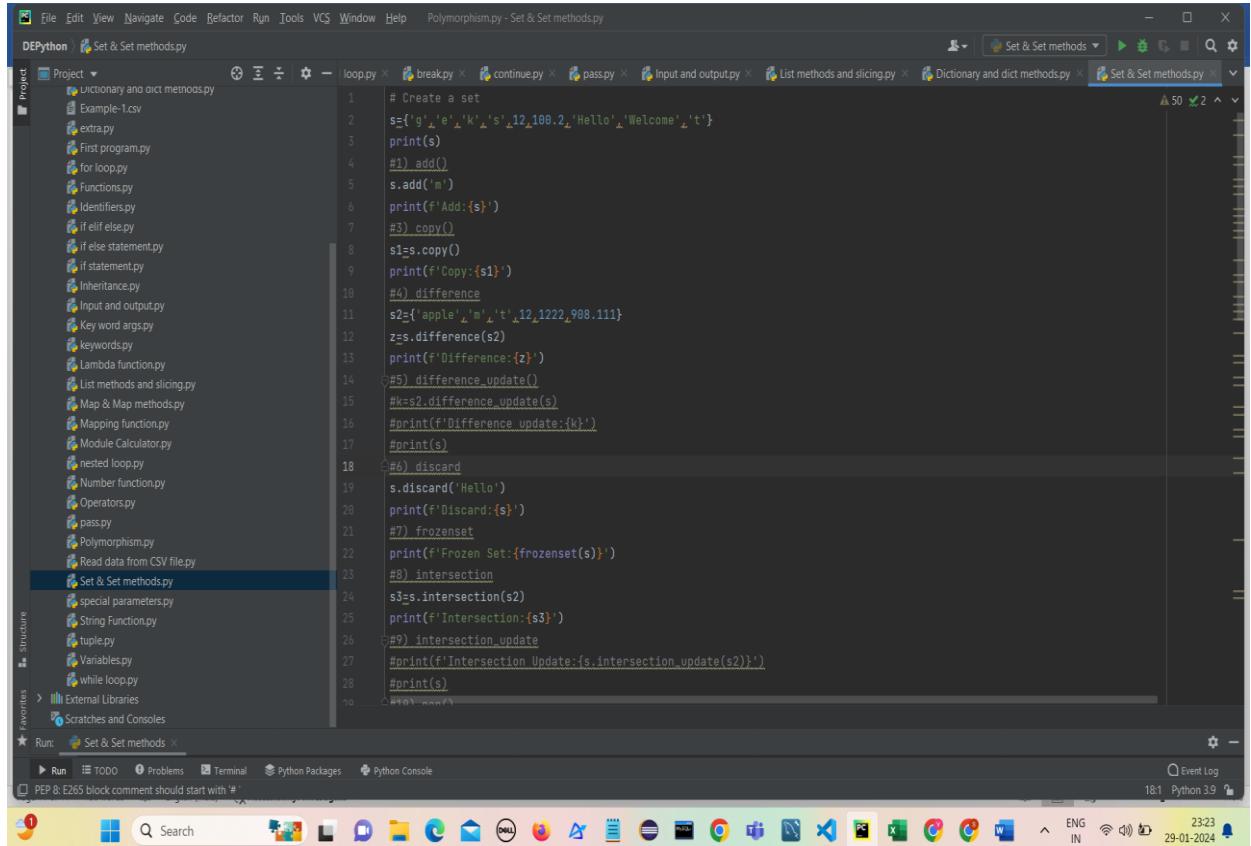
The run terminal shows the output of the code:

```
103 somya  
106 Kartik  
101 amit  
102 sumit  
103 somya  
106 Kartik  
{101: 'Salman', 102: 'sumit', 103: 'somya', 106: 'Kartik'}  
{101: 'Salman', 102: 'sumit', 103: 'somya', 106: 'Kartik', 105: 'kiran'}  
dict_keys([101, 102, 103, 104, 99, 100])  
dict_values(['Amit', 'Sumit', 'Somya', 'Sarthak', 'SAMI', 'sami'])  
dict_items([(101, 'Amit'), (102, 'Sumit'), (103, 'Somya'), (104, 'Sarthak'), (99, 'SAMI'), (100, 'sami')])  
Sumit  
{102: 'Sumit', 103: 'Somya', 104: 'Sarthak', 99: 'SAMI', 100: 'sami'}  
{102: 'Sumit', 103: 'Somya', 104: 'Sarthak', 99: 'SAMI'}  
Process finished with exit code 0
```

19) Introduction to Set and Set methods

```
# Set methods
'''

1) add()--Adds a given element to a set
2) clear()--Removes all elements from the set
3) copy()--Returns a shallow copy of the set
4) difference()--Returns a set that is the difference between two
sets(returns first set without common elements)
5) difference_update()--Updates the existing caller set with the difference
between two sets(removes items existing in both sets)
6) discard()--Removes the element from the set
7) frozenset()--Return an immutable frozenset object
8) intersection()--Returns a set that has the intersection of all sets
9) intersection_update()--Updates the existing caller set with the
intersection of sets
10) pop()--Returns and removes a random element from the set
11) remove()--Removes the element from the set
12) symmetric_difference()--returns a set that contains all items from both
set, but not the items that are present in both sets.
13) symmetric_difference_update()--updates the original set by removing
items that are present in both sets, and inserting the other items.
14) union()--Returns a set that has the union of all sets
15) update()--updates the current set, by adding items from another set
16) issubset()--Returns True if all elements of a set A are present in
another set B
17) issuperset()--Returns True if all elements of a set A occupies set B
'''
```



```
# Create a set
s={1,'e','K','s',12,100.2,'Hello','t'}
print(s)

#1) add()
s.add('m')
print(f'Add:{s}')

#3) copy()
s1=s.copy()
print(f'Copy:{s1}')

#4) difference
s2={'apple','m','t',12,1222,988.111}
z=s.difference(s2)
print(f'Difference:{z}')

#5) difference_update()
#s2.difference_update(s)
#print(f'Difference_update:{s}')
#print(s)

#6) discard
s.discard('Hello')
print(f'Discard:{s}')

#7) frozenset
print(f'Frozen Set:{frozenset(s)}')

#8) intersection
s3=s.intersection(s2)
print(f'Intersection:{s3}')

#9) intersection_update
#print(f'Intersection Update:{s.intersection_update(s2)}')

#print(s)
```

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - Set & Set methods.py. The left sidebar has sections for Project, Structure, Favorites, and Run. The main editor window displays Python code related to set operations. The code includes various methods like intersection_update, pop, remove, symmetric_difference, union, update, issubset, and issuperset. The status bar at the bottom shows the date as 29-01-2024 and the time as 18:1 Python 3.9.

```

25     print(f'Intersection:{s3}')
26     #9) intersection_update
27     #print(f'Intersection Update:{s.intersection_update(s2)}')
28     #print(s)
29     #10) pop()
30     print(f'Pop:{s.pop()}' )
31     #11) remove
32     print(f'Remove:{s2.remove(908,111)}')
33     print(s2)
34     #12) symmetric_difference
35     s4=s.symmetric_difference(s2)
36     print(f'Symmetric Difference:{s4}')
37     #13) symmetric_difference_update
38     x=s.symmetric_difference_update(s2)
39     print(f'Symmetric Difference Update:{x}')
40     print(s)
41     #14) union
42     y=s.union(s2)
43     print(f'Union:{y}')
44     #15) Update
45     s7={ 'a','b','c'}
46     s.update(s7)
47     print(f'Update:{s}')
48     #16) issubset()
49     print(f'Subset:{s1.issubset(s2)}')
50     #17) issuperset()
51     print(f'Subset:{s1.issuperset(s2)}')
52
#-*- coding: utf-8

```

Output:

The screenshot shows the PyCharm IDE interface with the output tab selected. The terminal window displays the execution of the script and its output. The output shows various set operations being performed on the string 'Hello'. The status bar at the bottom shows the date as 29-01-2024 and the time as 23:24 Python 3.9.

```

C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Set & Set methods.py"
[100, 2, 'Hello', 12, 'e', 'Welcome', 's', 'k', 'g', 't']
Add:{100, 2, 'Hello', 'm', 12, 'e', 'Welcome', 's', 'k', 'g', 't'}
Copy:{100, 2, 'Hello', 'm', 12, 'e', 'Welcome', 's', 'k', 'g', 't'}
Difference:{100, 2, 'Hello', 'e', 'Welcome', 's', 'k', 'g', 't'}
Discard:{100, 2, 'm', 12, 'e', 'Welcome', 's', 'k', 'g', 't'}
Frozen Set:frozenset({100, 2, 'm', 12, 'e', 'Welcome', 's', 'k', 'g', 't'})
Intersection:{'m', 't', 12}
Pop:100
Remove:None
{'apple', 1222, 'm', 't', 12}
Symmetric Difference:{'apple', 1222, 'g', 'e', 'Welcome', 's', 'k'}
Symmetric Difference Update:None
{'apple', 1222, 'e', 'Welcome', 's', 'k', 'g'}
Union:{'apple', 1222, 'm', 12, 'e', 'Welcome', 's', 'k', 'g', 't'}
Update:{1, 'apple', 2, 1222, 'e', 'b', 'Welcome', 's', 'k', 'a', 'g'}
Subset:False
Subset:False

Process finished with exit code 0

```

20) Map and Map methods

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - Map & Map methods.py. The left sidebar shows a project structure with files like continue.py, pass.py, Input and output.py, List methods and slicing.py, Dictionary and dict methods.py, Set & Set methods.py, and Map & Map methods.py. The main code editor contains the following Python script:

```
'''Map in Python is a function that works as an iterator to return a result after applying a function to every item of an iterable (tuple, lists, etc.). It is used when you want to apply a single transformation function to all the iterable elements.'''  
#map(function,iterables)  
# 1) Square of a number using map  
def square(i):  
    return i*i  
x=map(square,(2,3,4,5,6,7))  
print(x)  
print(list(x))  
  
#OR  
num=(11,12,13,14,15)  
result=map(square,num)  
output=list(result)  
print(output)  
  
# 2) Using Map() With Len()  
examples=['welcome','to','Python','Programming']  
x=map(len,examples)  
print(f'Length of example:{list(x)}')
```

The Run tab shows the output: [4, 9, 16, 25, 36, 49] [121, 144, 169, 196, 225]. The bottom status bar indicates the process finished with exit code 0 at 11:15 Python 3.9.

Output:

The screenshot shows the PyCharm IDE interface, identical to the previous one, but with the output displayed in the Run tab. The output is identical to the one shown in the first screenshot: [4, 9, 16, 25, 36, 49] [121, 144, 169, 196, 225]. The bottom status bar indicates the process finished with exit code 0 at 11:15 Python 3.9.

Functions

21) Mapping Function

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar says "Polymorphism.py - Mapping function.py". The main window displays the code for "Mapping function.py". The code uses the map function to calculate square roots and powers of numbers. The run tab at the bottom shows the output of the script.

```
#Square root using mapping function
import math
def sqrt(i):
    return math.sqrt(i)

res=map(sqrt,(16,256,1221,625))
print(res)
print("Square root:{List(res)}")

# Power of a number
powers=[1,2,3,4]
bases=[1,2,3,4]
result=list(map(pow,bases,powers))
print("Power of a number:{result}")
```

Output from the Run tab:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Mapping function.py"
<map object at 0x0000016910A62FA0>
Square root:[4.0, 16.0, 34.642810419312295, 25.0]
Power of a number:[1, 4, 27, 256]

Process finished with exit code 0
```

22) String Function

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar says "Polymorphism.py - String Function.py". The main window displays the code for "String Function.py". The code demonstrates various string methods like startswith, upper, lower, strip, capitalize, title, isalpha, isdigit, replace, and find. The run tab at the bottom shows the output of the script.

```
# String functions
s='pythonprogramming'
print(type(s))
print(s[3])
print(dir(s))
l=[]
for i in dir(s):
    if i.startswith('__'):
        pass
    else:
        l.append(i)
print(l)

print(s[:])
print(s[:-1])
print(s[5:-2])
print(s.upper())
print(s.lower())
print(s.strip())
print(s.capitalize())
print(s.title())
print(s.isalpha())# if spaces are present it returns false
string='12345'
print(string.isdigit())
print(s.startswith('p'))
print(s.endswith('g'))
print(s.replace('p','P'))
print(s.find('i'))# returns index of first occurrence
print(s.rfind('i'))# returns char with high value
for i in s:
    print(i) # if start with _
```

Output from the Run tab:

```
9:13 Python 3.9
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/String Function.py"
<map object at 0x0000016910A62FA0>
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__str__', '__subclasshook__', 'capitalize', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isdecimal', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'ljust', 'lower', 'lstrip', 'rfind', 'rindex', 'rjust', 'replace', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'upper', 'zfill']

Process finished with exit code 0
```

```

DEPython > String Function.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - String Function.py
Project ▾ 22 2
Dictionary and dict methods.py
Example-1.csv
extra.py
First program.py
for loop.py
Functions.py
Identifiers.py
if elif else.py
if else statement.py
if statement.py
Inheritance.py
Input and output.py
Key word args.py
Keywords.py
Lambda function.py
List methods and slicing.py
Map & Map methods.py
Mapping function.py
Module Calculator.py
Nested loop.py
Number function.py
Operators.py
pass.py
Polymorphism.py
Read data from CSV file.py
Set & Set methods.py
special parameters.py
String Function.py
tuple.py
Variables.py
while loop.py
External Libraries
Scratches and Consoles
Run: Mapping function
for i in dir(s): if i.startswith('__')
Run TODO Problems Terminal Python Packages Python Console
Event Log 9:13 Python 3.9
9:13 23:29 29-01-2024

```

Output:

```

DEPython > String Function.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - String Function.py
Project ▾ 22 2
Dictionary and dict methods.py
Example-1.csv
Run: String Function
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/String Function.py"
<class 'str'>
h
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__iadd__', '__imul__', '__init__', '__le__', '__lshift__', '__lt__', '__mul__', '__ne__', '__radd__', '__rdiv__', '__rmod__', '__rpow__', '__rsub__', '__rtruediv__', '__str__', '__sub__', '__truediv__']
['capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
pythonprogramming
gnimmargoronohtyp
mmi
PYTHONPROGRAMMING
pythonprogramming
pythonprogramming
Pythonprogramming
Pythonprogramming
True
True
True
True
PythonProgramming
14
y
a
['pyho', 'programmi', 'g']
Delhi
agghimnnnooprrty
gnimmargoronohtyp
gnimmargoronohtypgnimmargoronohtyp

Process finished with exit code 0
Run TODO Problems Terminal Python Packages Python Console
Event Log 23:30 Python 3.9
23:30 29-01-2024

```

23) Number Function

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - Number function.py. The main window displays a code editor with the following content:

```
#Math module
import math
print(math.pi)
print(math.floor(10.2))
print(math.ceil(10.2))
print(math.exp(2))
print(math.pow(10,2))
print(math.factorial(10))
print(math.sqrt(10.2))
```

The Project tool window on the left lists various Python files such as Example-1.csv, extra.py, First program.py, for loop.py, Functions.py, Identifiers.py, If elif else.py, If else statement.py, If statement.py, Inheritance.py, Input and output.py, Key word args.py, Keywords.py, Lambda functions.py, List methods and slicing.py, Map & Map methods.py, Mapping function.py, and Number function.py.

The Run tool window at the bottom shows the output of the script:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Number function.py"
3.141592653589793
10
11
7.38905609893065
100.0
3628800
3.1937438845342623

Process finished with exit code 0
```

The taskbar at the bottom of the screen shows various application icons and system status indicators.

24) Date and Time Function

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - Date and time function.py. The main window displays a code editor with the following content:

```
# date and time functions
import time
print(time.time())
print(time.localtime())
print(time.localtime().tm_year)
print(time.localtime().tm_mon)
print(time.localtime().tm_mday)
print(time.localtime().tm_hour)
print(time.asctime()) # formatted time which is readable format

# calendar
import calendar
print(calendar.month(2024,1))
```

The Project tool window on the left lists various Python files such as Constructor.py, continue.py, Datatypes.py, Date and time function.py, Default arguments.py, Dictionary and dict methods.py, Example-1.csv, extra.py, First program.py, for loop.py, Functions.py, Identifiers.py, If elif else.py, If else statement.py, If statement.py, and Inheritance.py.

The Run tool window at the bottom shows the output of the script:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Date and time function.py"
1706551325.277282
time.struct_time(tm_year=2024, tm_mon=1, tm_mday=29, tm_hour=23, tm_min=32, tm_sec=5, tm_wday=0, tm_yday=29, tm_isdst=0)
2024
 1
 29
 31
Mon Jan 29 23:32:05 2024
  January 2024
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

The taskbar at the bottom of the screen shows various application icons and system status indicators.

25) Functions

```
# Functions
def greetings():
    print('Welcome to Python')
greetings()

def sum():
    print('Enter two numbers')
    a, b = map(int, input().split())
    return a+b
ksum()
print(f'Sum={k}')

# or
def sum():
    print('Enter two numbers')
    a, b = map(int, input().split())
    c=a+b
    print(f"Sum={c}")
sum()

# a) Positional arguments
def fun(a,b):
    print(a,b)
fun(10,10)
# b) default arguments
def fun(a=0):
    print(a)
fun()
fun(10)

# with argument with return type
def fun(a,b=10):
    print(a,b)
fun(999)
fun(10,789)
# c) keyword arguments
def fun(a,b):
    print(a,b)
fun(a=234,b=789)
# d) variable length positional arguments
def fun(*args):
    print(type(args))
    for i in args:
        print(i,end=' ')
fun(10,20,30,40,50)

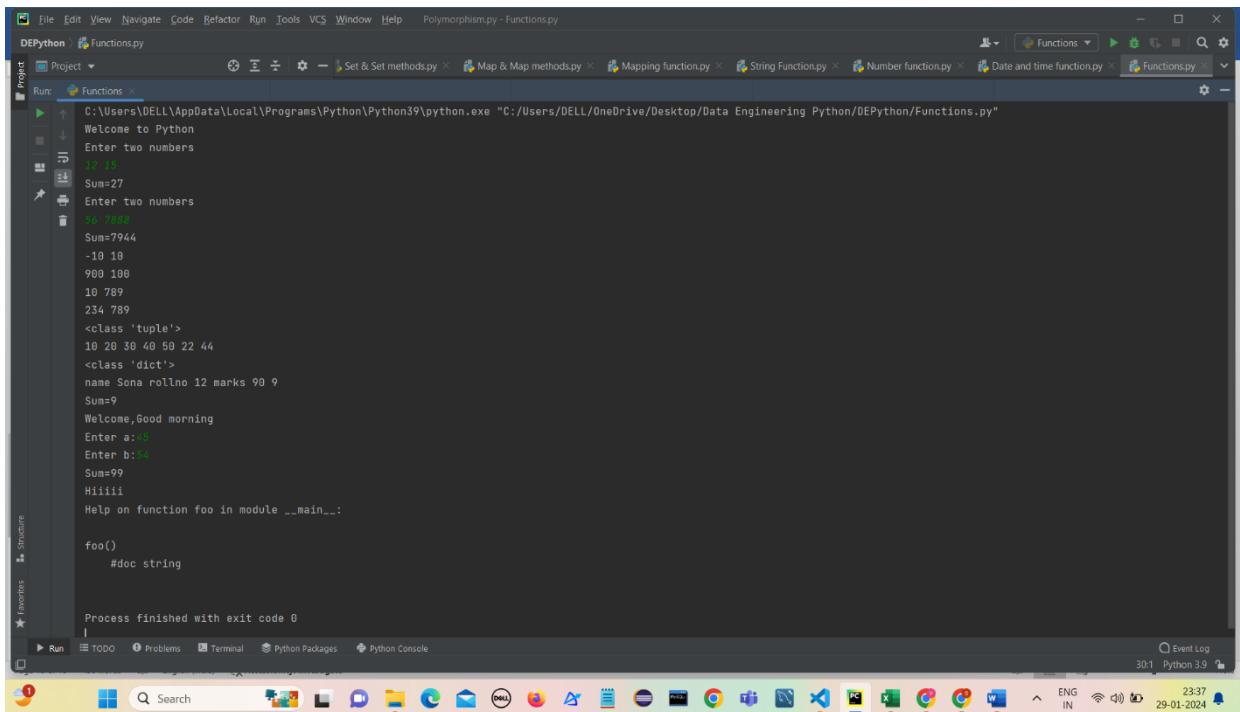
# or
def fun(*args):
    print(args[0],args[2])
fun(22,33,44,55,66)
# e) variable length keyword arguments
def fun(**kwargs):
    print(type(kwargs))
    for k,v in kwargs.items():
        print(k,v,end=' ')
fun(name='Sonu',rolling=12,marks=90)
```

```
# b) default arguments
def fun(a,b=10):
    print(a,b)
fun()
fun(999)
fun(10,789)
# c) keyword arguments
def fun(a,b):
    print(a,b)
fun(a=234,b=789)
# d) variable length positional arguments
def fun(*args):
    print(type(args))
    for i in args:
        print(i,end=' ')
fun(10,20,30,40,50)

# or
def fun(*args):
    print(args[0],args[2])
fun(22,33,44,55,66)
# e) variable length keyword arguments
def fun(**kwargs):
    print(type(kwargs))
    for k,v in kwargs.items():
        print(k,v,end=' ')
fun(name='Sonu',rolling=12,marks=90)
```

```
print(k,v,end=' ')
# with argument with return type
# without argument with return type
# without argument without return type
# without argument with return type
# with argument with return type
def example(a,b):
    return a+b
print(example(2,7))
# without argument with return type
def example(a):
    print(f'{a}+{b}={a+b}')
example(2)
# without argument with return type
def example():
    return "Welcome,Good morning"
print(example())
# without argument without return type
def example():
    a=int(input("Enter a:"))
    b=int(input("Enter b:"))
    print(f"sum={a+b}")
example()
executing
def foo():
    return 'Hello'
print(foo())
help(foo)
```

Output:

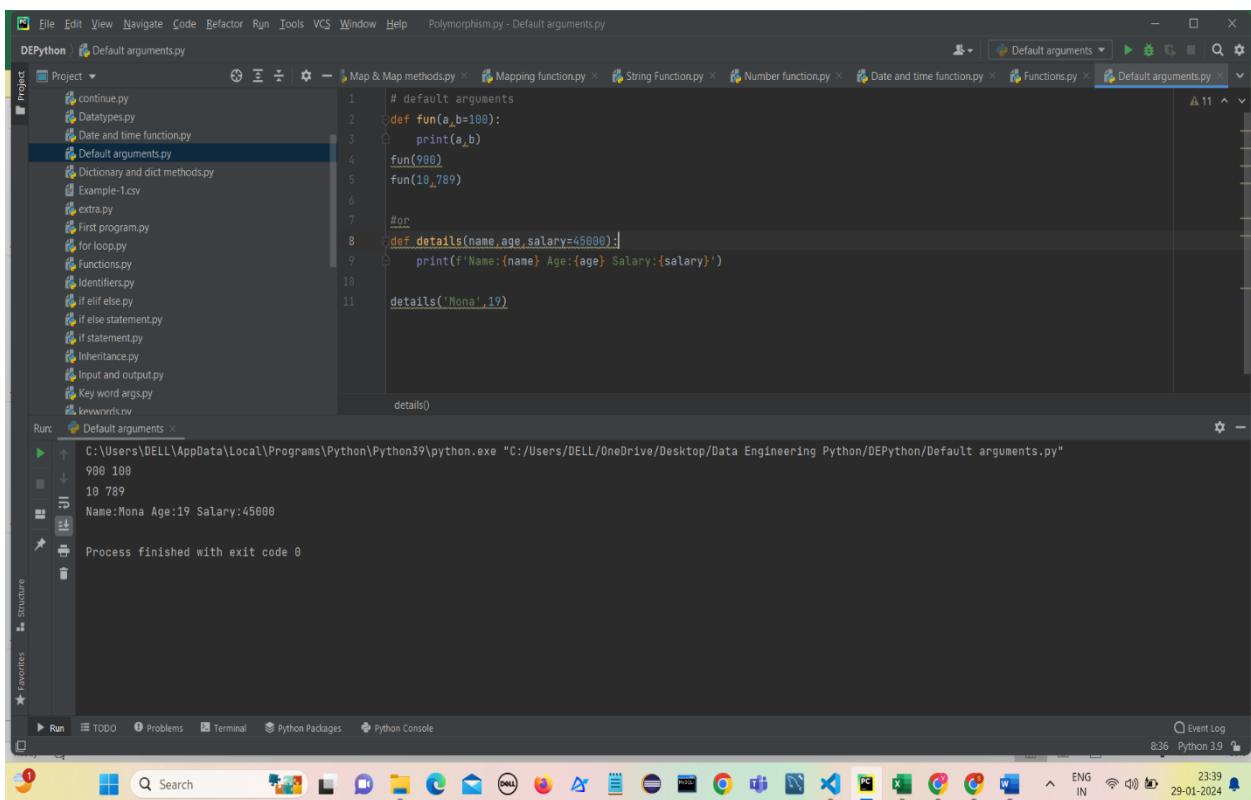


```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Functions.py"
Welcome to Python
Enter two numbers
10 10
Sum=20
Enter two numbers
50 100
Sum=150
-10 10
900 100
10 789
234 789
<class 'tuple'>
10 20 30 40 50 22 44
<class 'dict'>
name Sona rollno 12 marks 99 9
Sum=9
Welcome,Good morning
Enter a:44
Enter b:34
Sum=99
Hiiiii
Help on function foo in module __main__:

foo()
    #doc string

Process finished with exit code 0
```

26) Default Argument Values



```
# default arguments
def fun(a,b=100):
    print(a,b)
fun(980)
fun(10,789)

#or
def details(name,age,salary=45000):
    print(f'Name:{name} Age:{age} Salary:{salary}')
details('Mona',19)

Process finished with exit code 0
```

27) Keyword arguments

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists several Python files: extra.py, First program.py, for loop.py, Functions.py, Identifiers.py, if elif else.py, if else statement.py, Inheritance.py, Input and output.py, Key word args.py, keywords.py, Lambda function.py, List methods and slicing.py, Map & Map methods.py, Mapping function.py, Module Calculator.py, and nested loop.py. The current file, Key word args.py, is open in the editor. The code defines two functions: `fun(a,b)` which prints the values of `a` and `b`, and `details(name,age)` which prints a formatted string. The `details` function is called with two arguments: `details('Sona',22)` and `details('Mona',19)`. The run tab at the bottom shows the output: `Name:Sona,Age:22` and `Name:Mona,Age:19`. The status bar at the bottom right indicates Python 3.9 and the date 29-01-2024.

```
#keyword_arguments
def fun(a,b):
    print(a,b)
fun(a=234,b=789)

#or
def details(name,age):
    print(f'Name:{name},Age:{age}')

details('Sona',22)
details('Mona',19)
```

28) Special Parameters

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists various Python files, including special parameters.py, which is currently selected. The code in the editor illustrates the use of special parameters. It includes examples of `*args` and `**kwargs`. The `*args` example defines a function `def fruits(*args):` that prints each argument. The `**kwargs` example defines a function `def winner(**kwargs):` that prints key-value pairs from a dictionary. The run tab at the bottom shows the output: `Fruits:` followed by a list of fruit names (apple, mango, banana, berry), and `winner(First='Sona',Second='Mona',Third='Dinnu')`. The status bar at the bottom right indicates Python 3.9 and the date 29-01-2024.

```
'''Python have the special forms of parameters in functions.
These special forms of parameters are Args and Kwargs.
Args is a special parameter through which any number of positive arguments packed into a tuple.
Through Kwargs any number of Keyword Arguments packed into a dictionary.

# Args example
def fruits(*args):
    for i in args:
        print(i)

print("Fruits:")
fruits('apple','mango','banana','berry')

# Kwargs example
def winner(**kwargs):
    for k,v in kwargs.items():
        print(f'Key:{k} Value:{v}')

winner(First='Sona',Second='Mona',Third='Dinnu')
```

29) Arbitrary argument lists

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top lists files like Number function.py, Date and time function.py, Functions.py, Default arguments.py, Key word args.py, special parameters.py, and Arbitrary args.py. The main code editor window displays Python code for handling variable-length argument lists. The code includes examples for variable-length positional arguments using *args and variable-length keyword arguments using **kwargs. A run configuration for 'Arbitrary args' is selected in the bottom toolbar.

```
#Variable length positional arguments
def fun(*args):
    print(type(args))
    for i in args:
        print(i,end=' ')
fun(10,20,30,40,50)

# or
# sum of numbers
def add(*args):
    s=0
    for x in args:
        s+=x
    return s
result = add(10,20,30,40)
print(f'\nSum:{result}')

#Variable length keyword arguments
def address(**kwargs):
    for k,v in kwargs.items():
        print(f'{k}:{v}')
print('Variable length keyword arguments')
print('pass four keyword args')
address(Name="Raam", City="Mumbai", ph_no="9123134567", PIN="400001")
```

The screenshot shows the PyCharm IDE interface after running the code. The run output window at the bottom displays the execution results. It shows the output of the variable-length positional arguments (10, 20, 30, 40, 50), the sum of these numbers (150), and the variable-length keyword arguments (Name: Raam, City: Mumbai, ph_no: 9123134567, PIN: 400001). The status bar at the bottom right indicates the date and time as 29-01-2024 and 23:46.

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Arbitrary args.py"
<class 'tuple'>
10 20 30 40 50
Sum:150
Variable length keyword arguments
pass four keyword args
Name:Raam
City:Mumbai
ph_no:9123134567
PIN:400001

Process finished with exit code 0
```

30) Lambda Expressions

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Polymorphism.py - Lambda function.py. The left sidebar has a Project view with files like if else statement.py, if statement.py, Inheritance.py, Input and output.py, Key word args.py, keywords.py, Lambda function.py (which is selected), List methods and slicing.py, Map & Map methods.py, Mapping function.py, Module Calculator.py, nested loop.py, Number function.py, Operators.py, pass.py, Polymorphism.py, Read data from CSV file.py, Set & Set methods.py, special parameters.py, String Function.py, tuple.py, and Variables.py. The main code editor window displays Python code for lambda functions:

```
'''A lambda function is a small anonymous function.  
A lambda function can take any number of arguments, but can only have one expression.  
lambda arguments : expression'''  
  
##1) simple example  
print('Simple example using lambda function')  
greet=lambda:print("Hello,Welcome to Python")  
greet()  
  
##2) Lambda function with _arguments  
print("Lambda function with args")  
Details=lambda name,age:print(f'I am {name}, and I am {age} years old')  
Details("SriGanga",22)  
  
##3) Sum of two numbers  
print("Lambda function to find sum of two numbers")  
sum=lambda x,y:print(f'Sum={x+y}')  
sum(12,45)  
sum(1.55,54.98)
```

The Run tab shows the output of the Lambda function script:

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/Lambda function.py"  
Simple example using lambda function  
Hello,Welcome to Python  
Lambda function with args  
I am SriGanga and I am 22 years old  
Lambda function to find sum of two numbers  
Sum=57  
Sum=56.52999999999999
```

The bottom status bar shows the system tray icons and the date/time.