
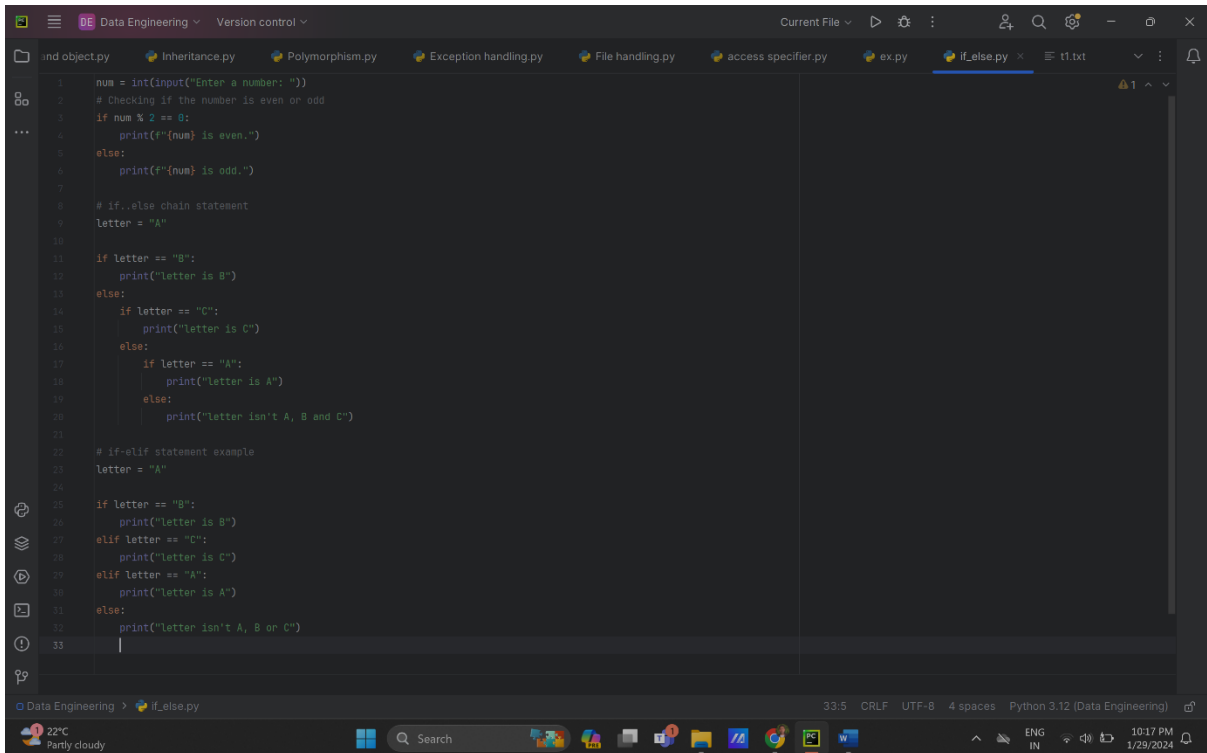


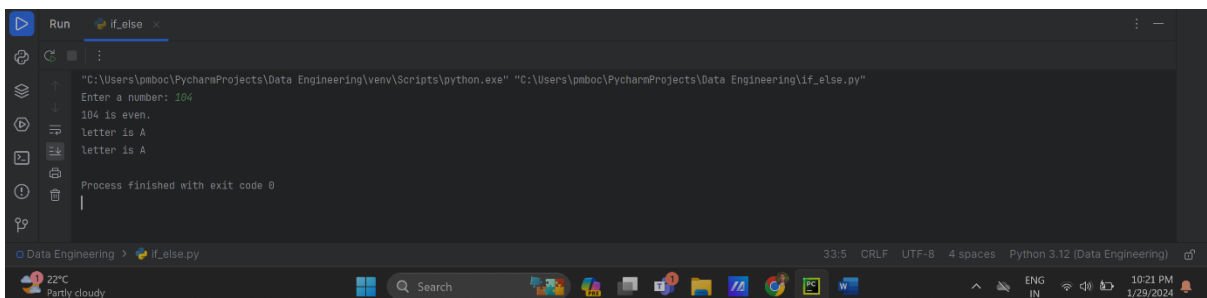
Name: Pradip Bochare

 If, If-else, if-elif-else statement



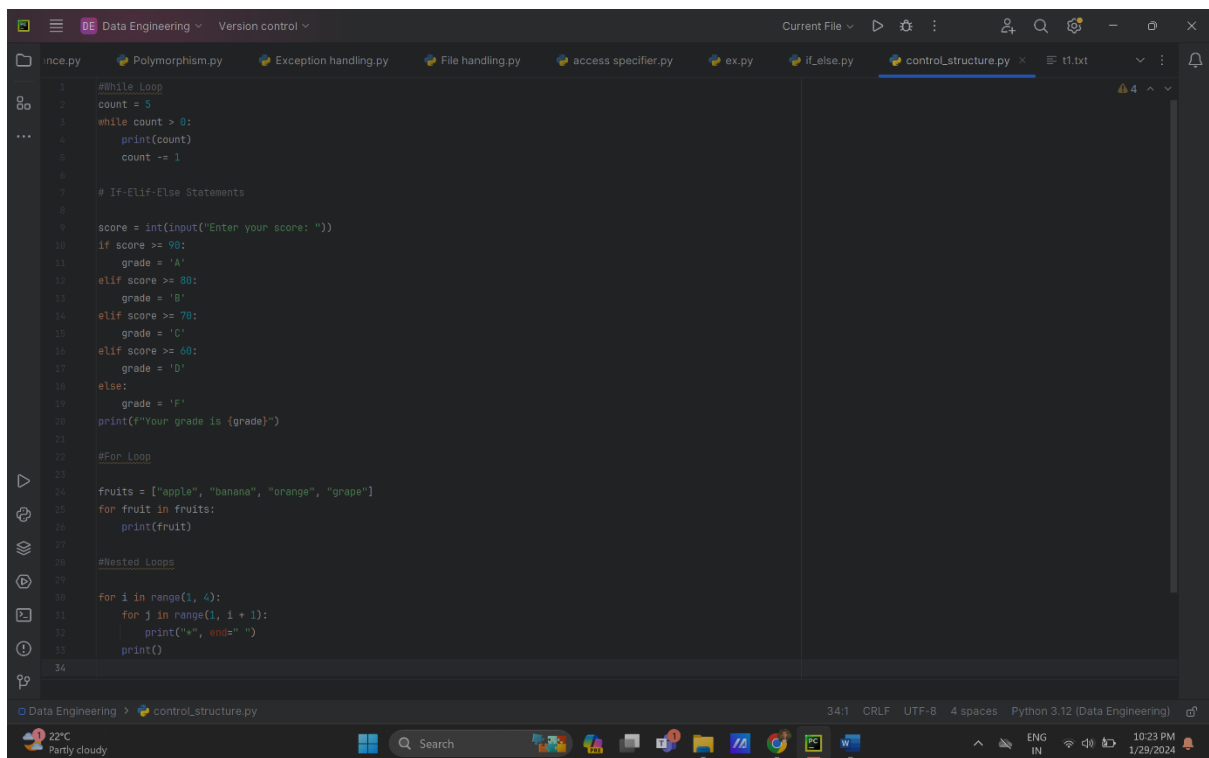
```
1 num = int(input("Enter a number: "))
2 # Checking if the number is even or odd
3 if num % 2 == 0:
4     print(f"{num} is even.")
5 else:
6     print(f"{num} is odd.")
7
8 # if..else chain statement
9 letter = "A"
10
11 if letter == "B":
12     print("letter is B")
13 else:
14     if letter == "C":
15         print("letter is C")
16     else:
17         if letter == "A":
18             print("letter is A")
19         else:
20             print("letter isn't A, B and C")
21
22 # if-elif statement example
23 letter = "A"
24
25 if letter == "B":
26     print("letter is B")
27 elif letter == "C":
28     print("letter is C")
29 elif letter == "A":
30     print("letter is A")
31 else:
32     print("letter isn't A, B or C")
33
```

Result



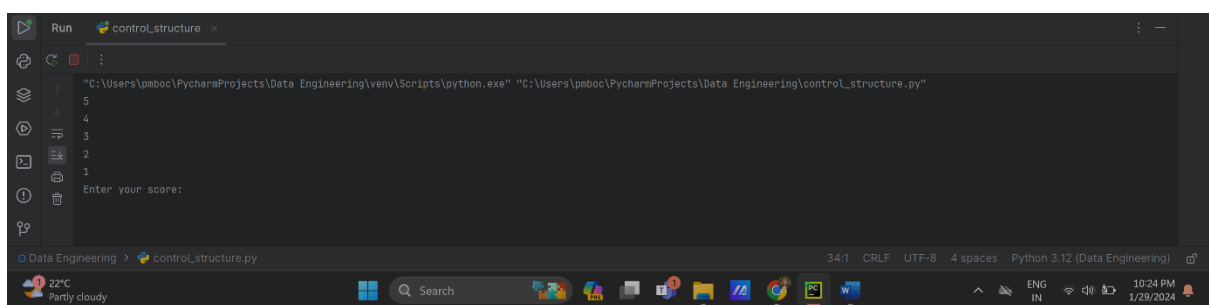
```
Run if_else.py
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\if_else.py"
Enter a number: 104
104 is even.
letter is A
letter is A
Process finished with exit code 0
```

Control Structure



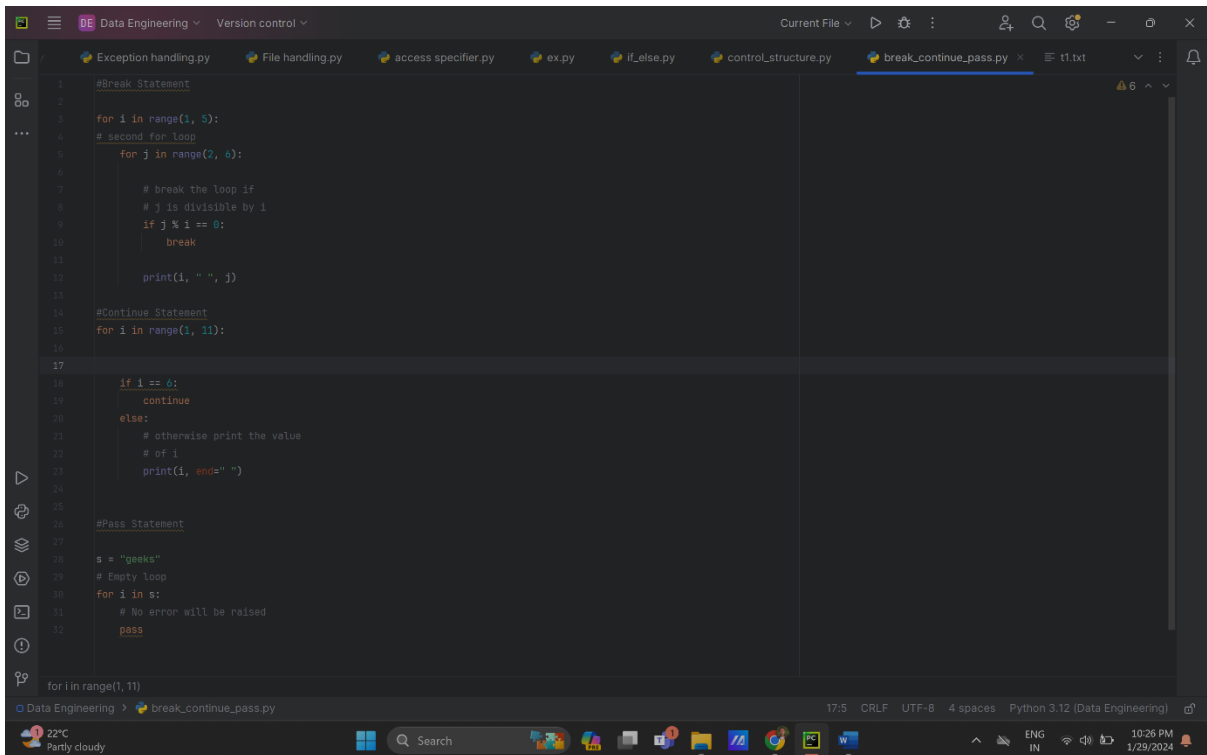
```
1 #While Loop
2 count = 5
3 while count > 0:
4     print(count)
5     count -= 1
6
7 # If-Elif-Else Statements
8
9 score = int(input("Enter your score: "))
10 if score >= 90:
11     grade = 'A'
12 elif score >= 80:
13     grade = 'B'
14 elif score >= 70:
15     grade = 'C'
16 elif score >= 60:
17     grade = 'D'
18 else:
19     grade = 'F'
20 print(f"Your grade is {grade}")
21
22 #For Loop
23
24 fruits = ["apple", "banana", "orange", "grape"]
25 for fruit in fruits:
26     print(fruit)
27
28 #Nested Loops
29
30 for i in range(1, 4):
31     for j in range(1, 1 + 1):
32         print("x", end=" ")
33     print()
34
```

Result



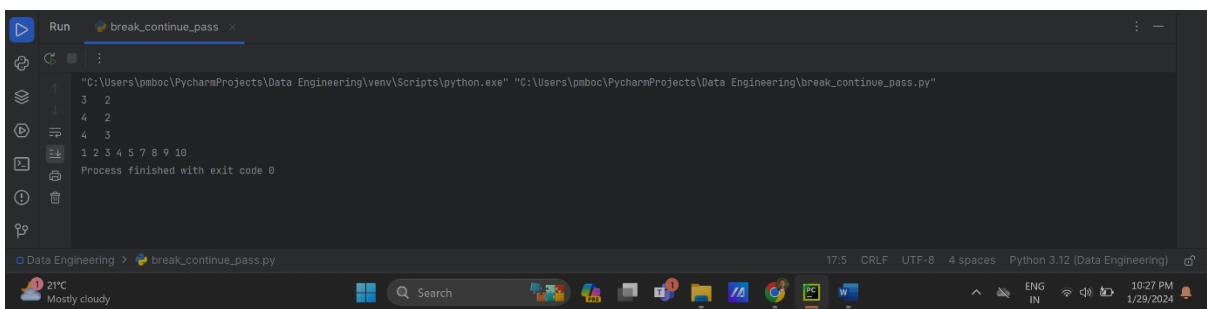
```
Run control_structure x
C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe "C:\Users\pmboc\PycharmProjects\Data_Engineering\control_structure.py"
5
4
3
2
1
Enter your score:
```

Break, Continue, Pass



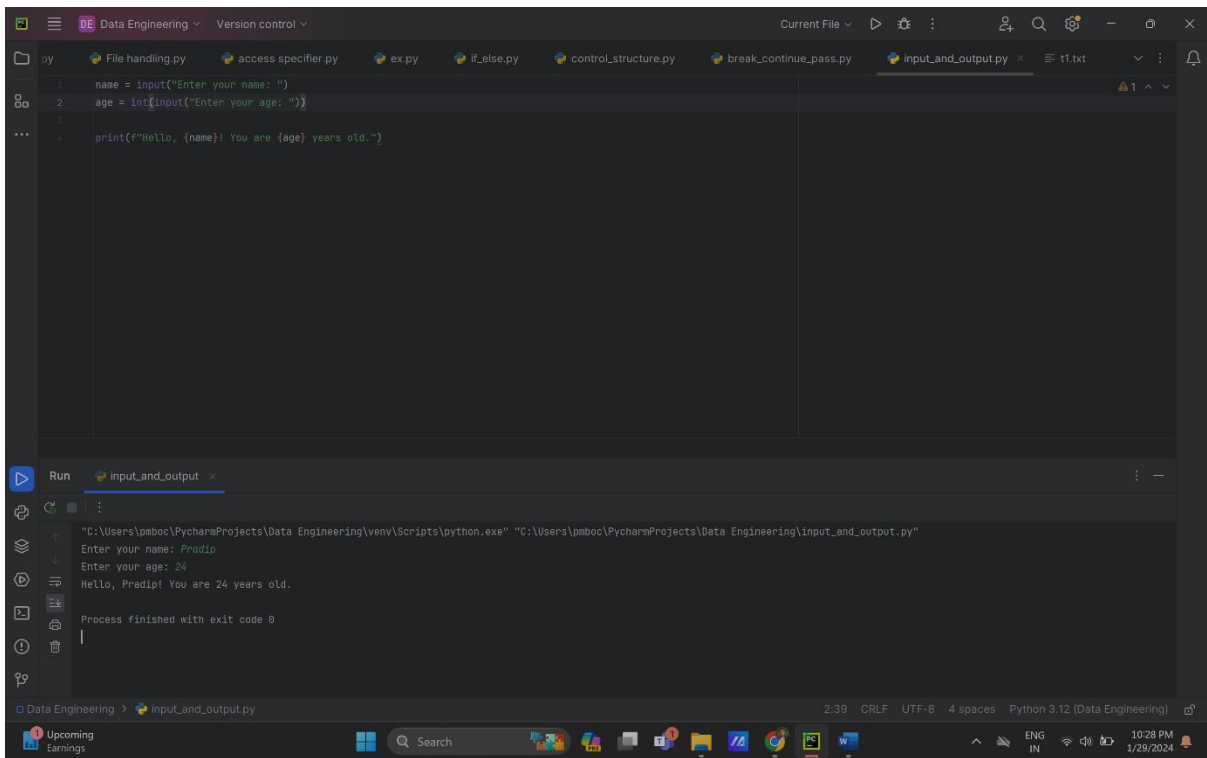
```
1 #Break Statement
2
3 for i in range(1, 5):
4     # second for loop
5     for j in range(2, 6):
6
7         # break the loop if
8         # j is divisible by i
9         if j % i == 0:
10             break
11
12     print(i, " ", j)
13
14 #Continue Statement
15 for i in range(1, 11):
16
17
18     if i == 6:
19         continue
20     else:
21         # otherwise print the value
22         # of i
23         print(i, end=" ")
24
25
26 #Pass Statement
27
28 s = "geeks"
29 # Empty loop
30 for i in s:
31     # No error will be raised
32     pass
33
34 for i in range(1, 11)
```

Result



```
Run break_continue_pass
C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe "C:\Users\pmboc\PycharmProjects\Data_Engineering\break_continue_pass.py"
3 2
4 2
4 3
1 2 3 4 5 7 8 9 10
Process finished with exit code 0
```

Input and Output



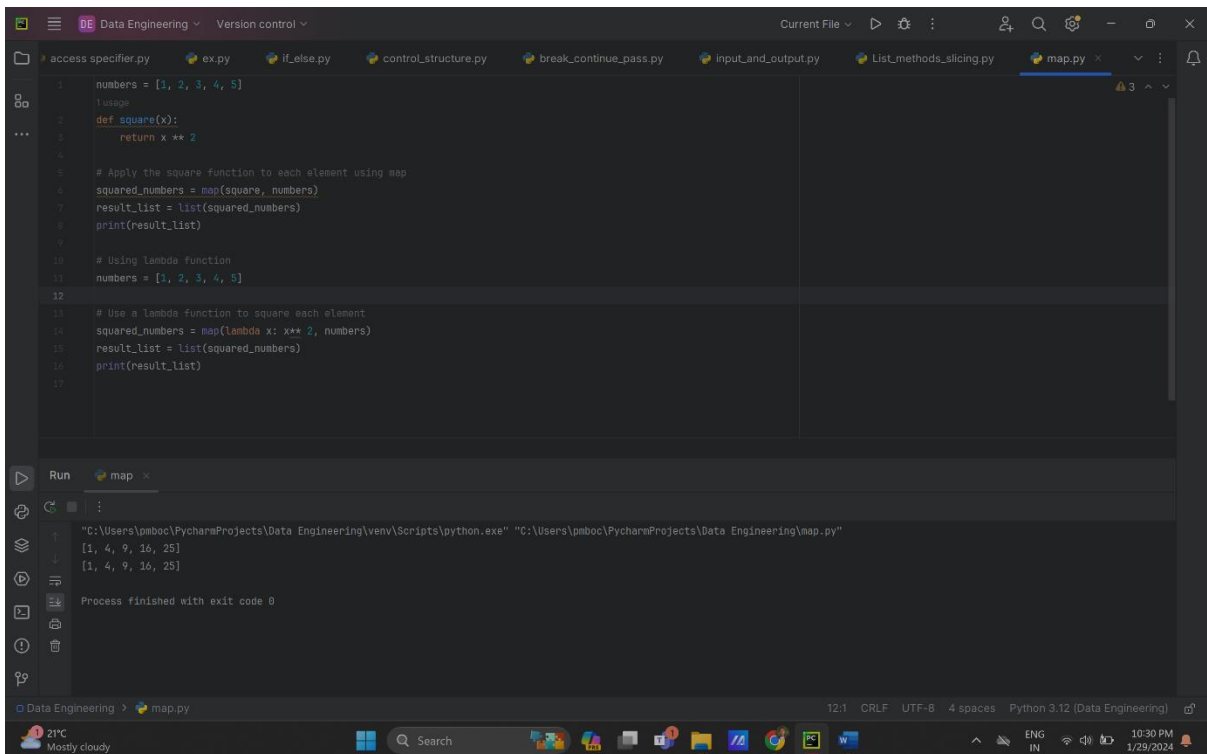
```
1 name = input("Enter your name: ")
2 age = int(input("Enter your age: "))
3
4 print(f"Hello, {name}! You are {age} years old.")
```

Run **input_and_output**

```
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\input_and_output.py"
Enter your name: Pradip
Enter your age: 24
Hello, Pradip! You are 24 years old.
Process finished with exit code 0
```

2:39 CRLF UTF-8 4 spaces Python 3.12 (Data Engineering)

Map Function



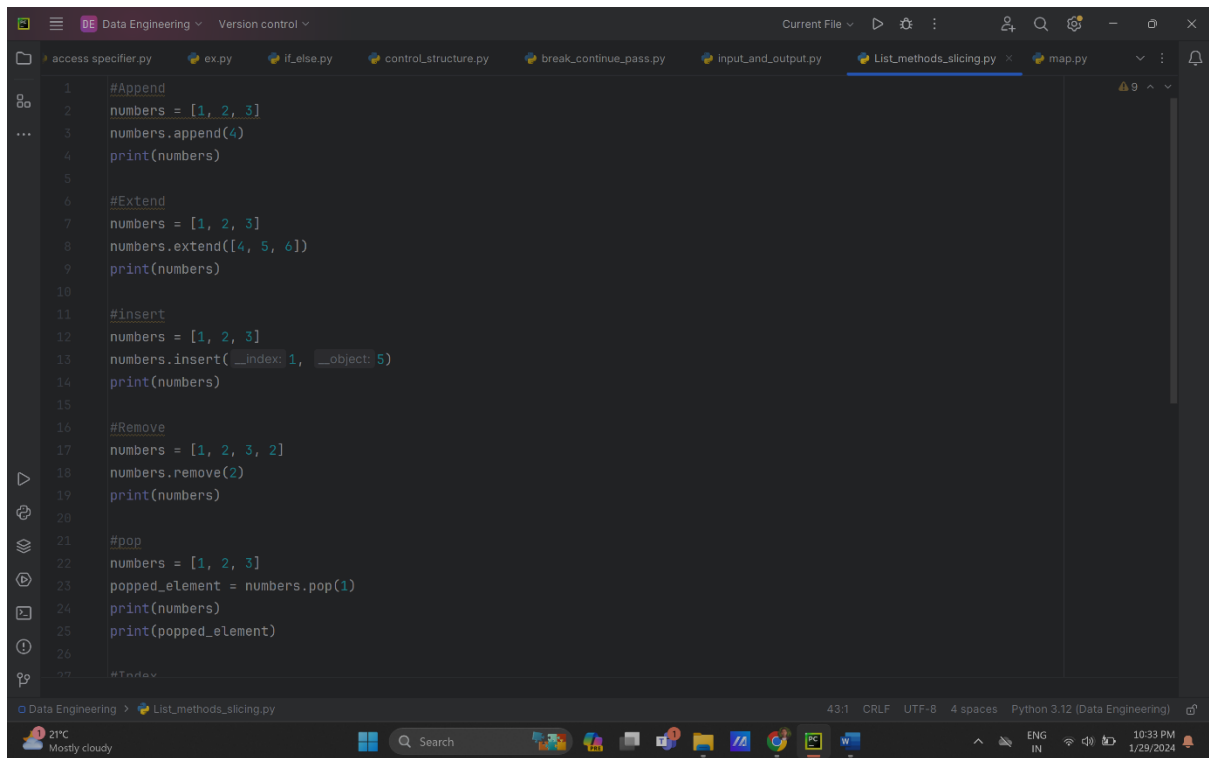
```
1 numbers = [1, 2, 3, 4, 5]
2 def square(x):
3     return x ** 2
4
5 # Apply the square function to each element using map
6 squared_numbers = map(square, numbers)
7 result_list = list(squared_numbers)
8 print(result_list)
9
10 # Using lambda function
11 numbers = [1, 2, 3, 4, 5]
12
13 # Use a lambda function to square each element
14 squared_numbers = map(lambda x: x ** 2, numbers)
15 result_list = list(squared_numbers)
16 print(result_list)
17
```

Run **map**

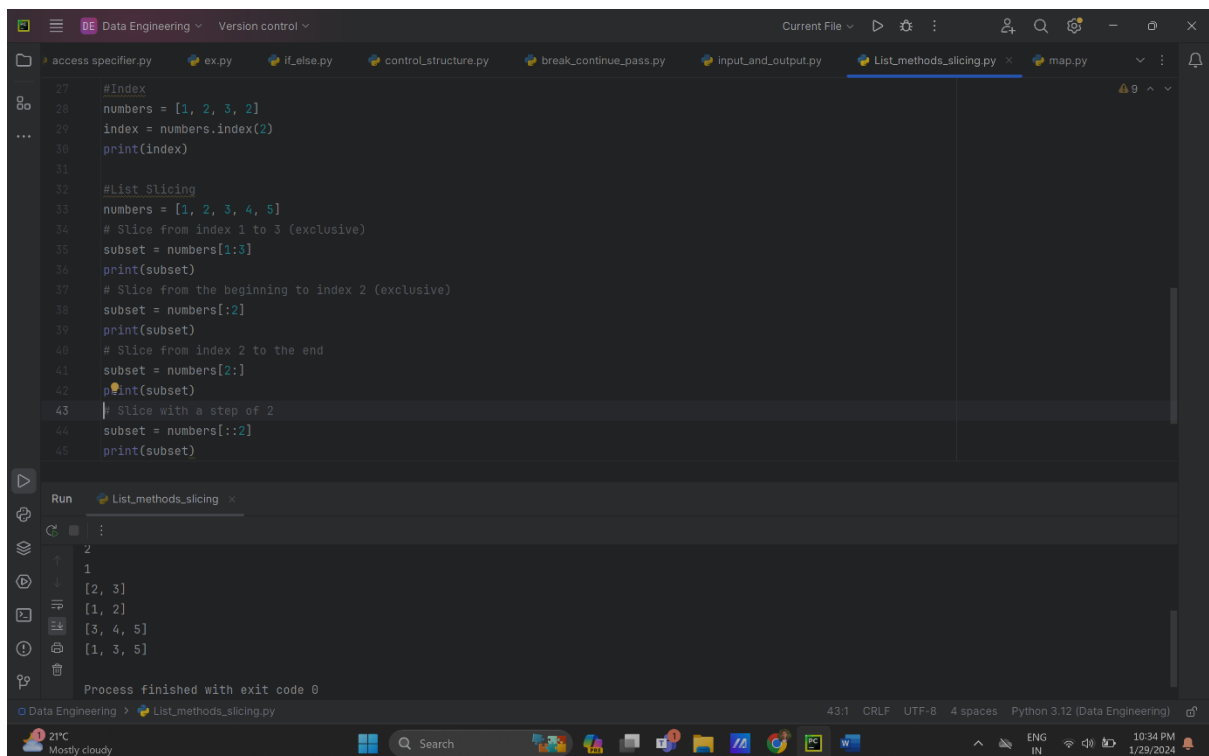
```
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\map.py"
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25]
Process finished with exit code 0
```

12:1 CRLF UTF-8 4 spaces Python 3.12 (Data Engineering)

List Methods and Slicing



```
1 #Append
2 numbers = [1, 2, 3]
3 numbers.append(4)
4 print(numbers)
5
6 #Extend
7 numbers = [1, 2, 3]
8 numbers.extend([4, 5, 6])
9 print(numbers)
10
11 #insert
12 numbers = [1, 2, 3]
13 numbers.insert(_index: 1, _object: 5)
14 print(numbers)
15
16 #Remove
17 numbers = [1, 2, 3, 2]
18 numbers.remove(2)
19 print(numbers)
20
21 #pop
22 numbers = [1, 2, 3]
23 popped_element = numbers.pop(1)
24 print(numbers)
25 print(popped_element)
26
27 #Task
```



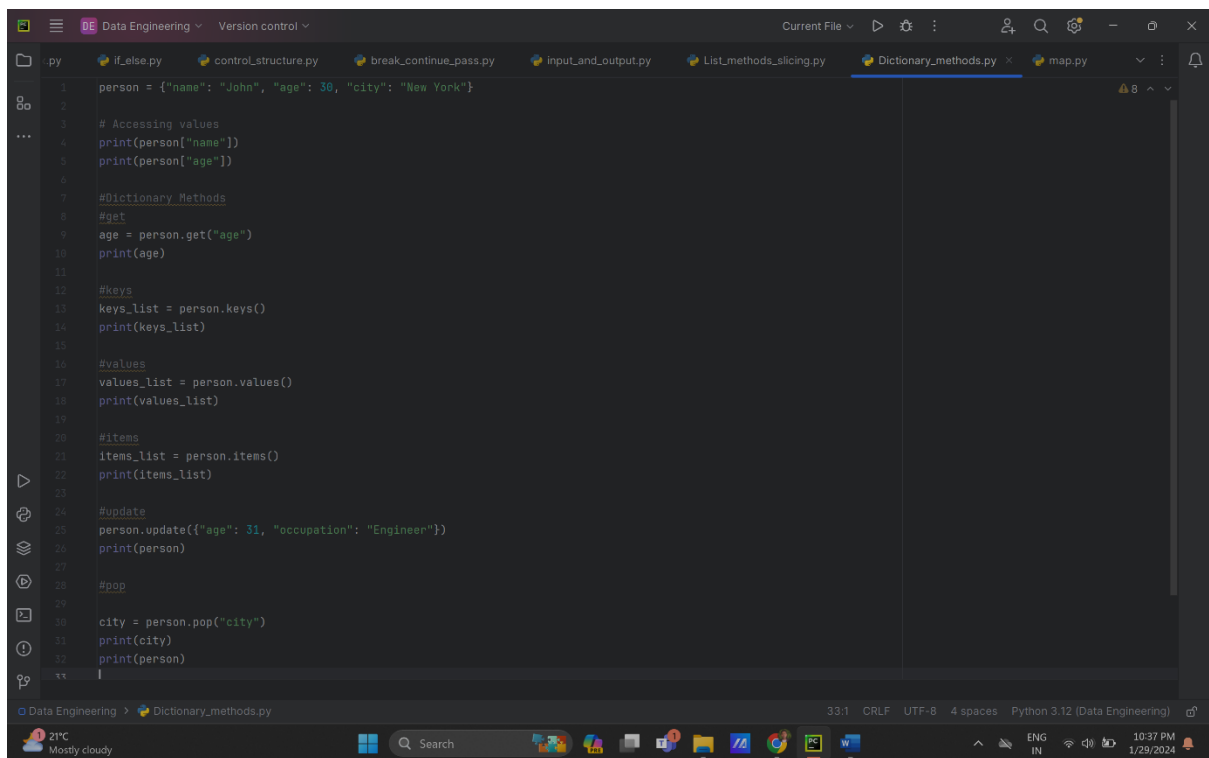
```
27 #Index
28 numbers = [1, 2, 3, 2]
29 index = numbers.index(2)
30 print(index)
31
32 #List Slicing
33 numbers = [1, 2, 3, 4, 5]
34 # Slice from index 1 to 3 (exclusive)
35 subset = numbers[1:3]
36 print(subset)
37 # Slice from the beginning to index 2 (exclusive)
38 subset = numbers[:2]
39 print(subset)
40 # Slice from index 2 to the end
41 subset = numbers[2:]
42 print(subset)
43 # Slice with a step of 2
44 subset = numbers[::2]
45 print(subset)
```

Run List_methods_slicing

```
2
1
[2, 3]
[1, 2]
[3, 4, 5]
[1, 3, 5]
```

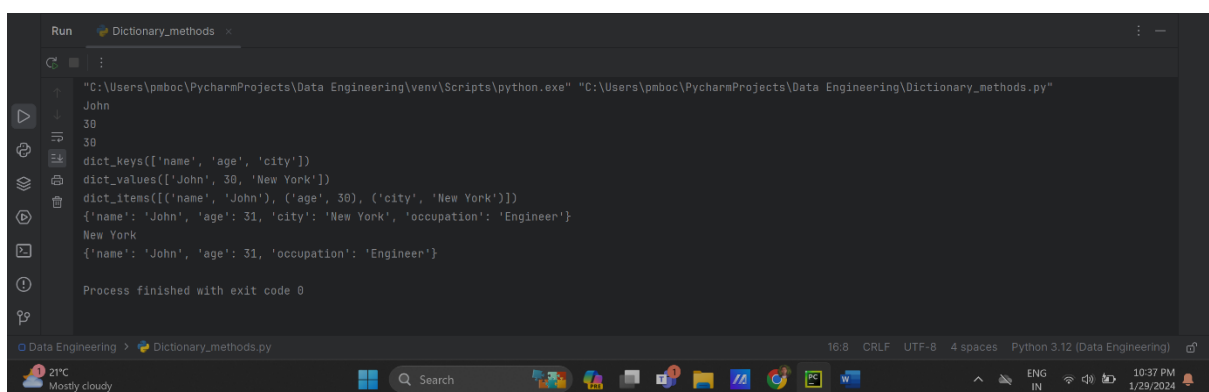
Process finished with exit code 0

Dictionaries and Dictionary Methods



```
1 person = {"name": "John", "age": 30, "city": "New York"}
2
3 # Accessing values
4 print(person["name"])
5 print(person["age"])
6
7 # Dictionary Methods
8 #get
9 age = person.get("age")
10 print(age)
11
12 #keys
13 keys_list = person.keys()
14 print(keys_list)
15
16 #values
17 values_list = person.values()
18 print(values_list)
19
20 #items
21 items_list = person.items()
22 print(items_list)
23
24 #update
25 person.update({"age": 31, "occupation": "Engineer"})
26 print(person)
27
28 #pop
29
30 city = person.pop("city")
31 print(city)
32 print(person)
33
```

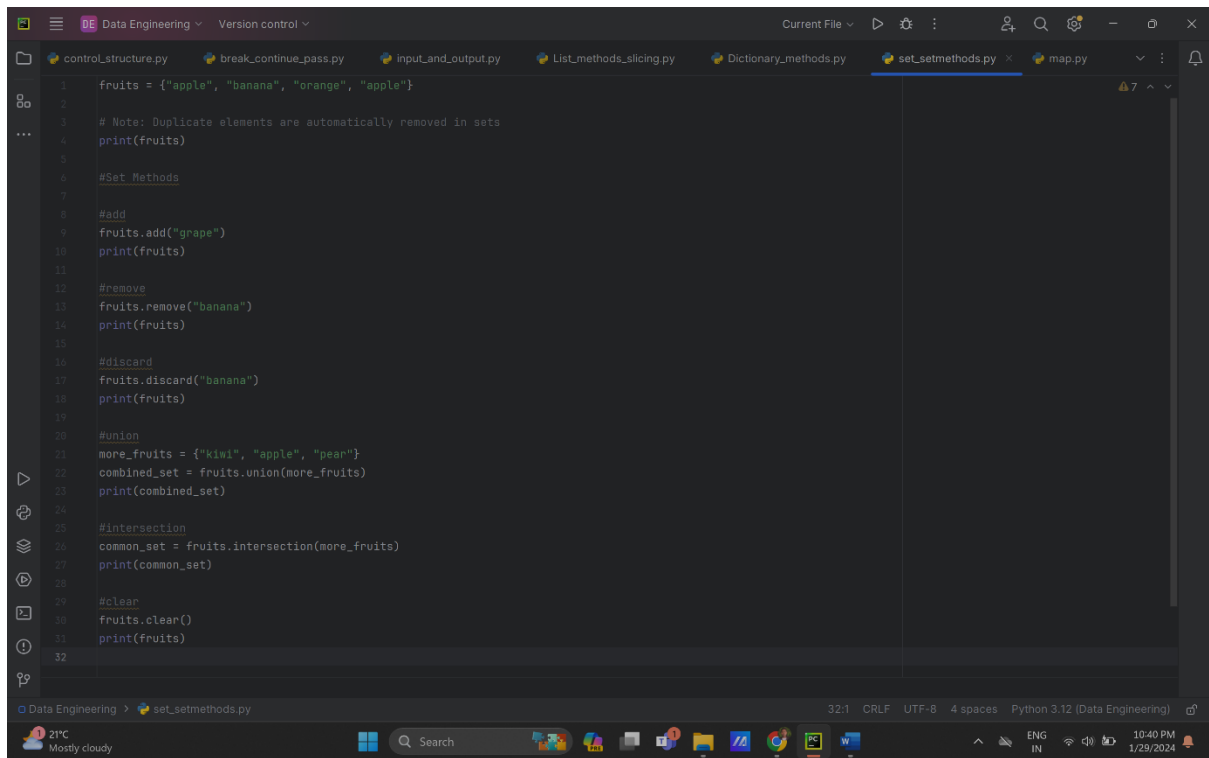
Result



```
Run Dictionary_methods
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\Dictionary_methods.py"
John
30
30
dict_keys(['name', 'age', 'city'])
dict_values(['John', 30, 'New York'])
dict_items([('name', 'John'), ('age', 30), ('city', 'New York')])
{'name': 'John', 'age': 31, 'city': 'New York', 'occupation': 'Engineer'}
New York
{'name': 'John', 'age': 31, 'occupation': 'Engineer'}

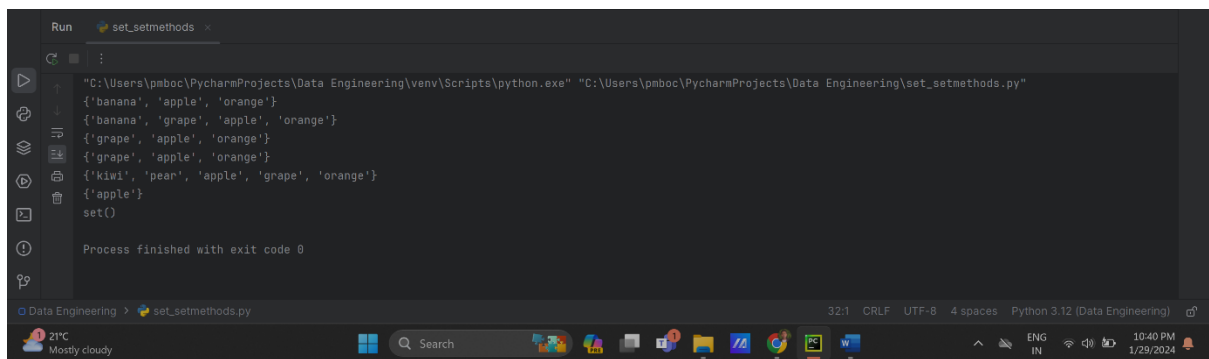
Process finished with exit code 0
```

Set and Set Methods



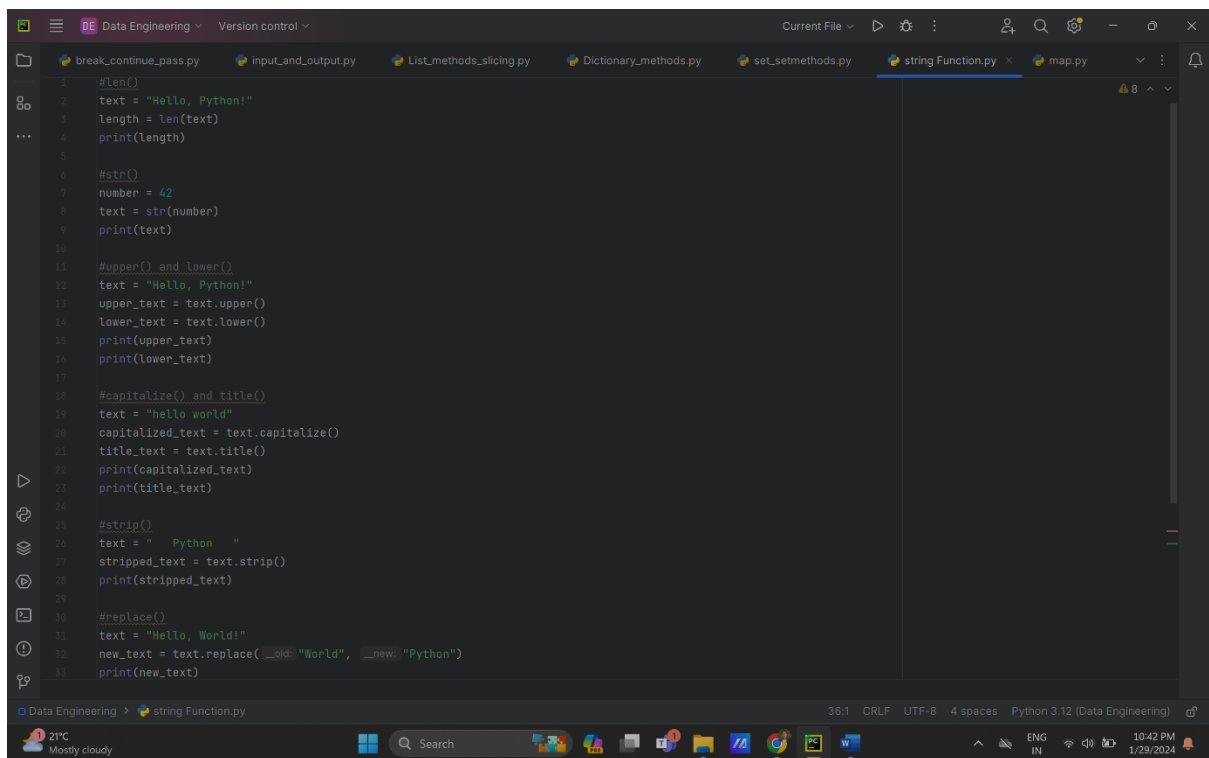
```
1 fruits = {"apple", "banana", "orange", "apple"}
2
3 # Note: Duplicate elements are automatically removed in sets
4 print(fruits)
5
6 #Set Methods
7
8 #add
9 fruits.add("grape")
10 print(fruits)
11
12 #remove
13 fruits.remove("banana")
14 print(fruits)
15
16 #discard
17 fruits.discard("banana")
18 print(fruits)
19
20 #union
21 more_fruits = {"kiwi", "apple", "pear"}
22 combined_set = fruits.union(more_fruits)
23 print(combined_set)
24
25 #intersection
26 common_set = fruits.intersection(more_fruits)
27 print(common_set)
28
29 #clear
30 fruits.clear()
31 print(fruits)
32
```

Result



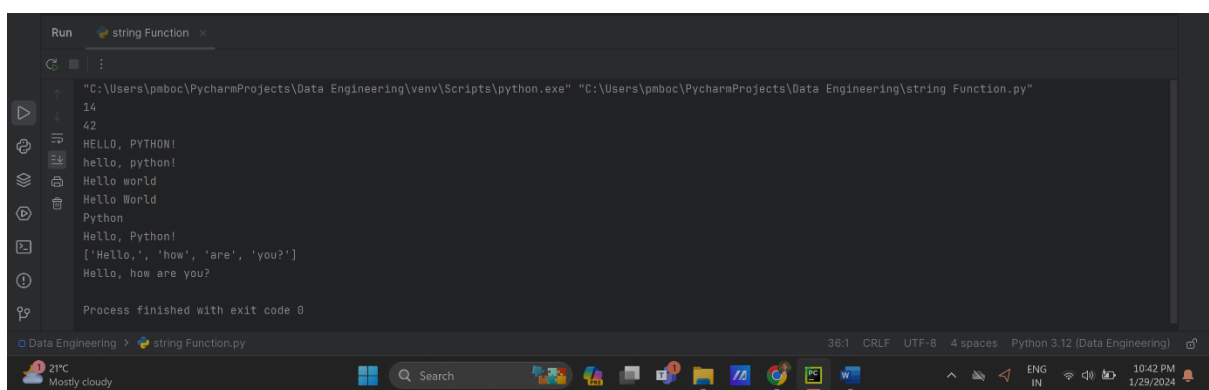
```
Run set_setmethods.py
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\set_setmethods.py"
{'banana', 'apple', 'orange'}
{'banana', 'grape', 'apple', 'orange'}
{'grape', 'apple', 'orange'}
{'grape', 'apple', 'orange'}
{'kiwi', 'pear', 'apple', 'grape', 'orange'}
{'apple'}
set()
Process finished with exit code 0
```

String Function



```
1 #len()
2 text = "Hello, Python!"
3 length = len(text)
4 print(length)
5
6 #str()
7 number = 42
8 text = str(number)
9 print(text)
10
11 #upper() and lower()
12 text = "Hello, Python!"
13 upper_text = text.upper()
14 lower_text = text.lower()
15 print(upper_text)
16 print(lower_text)
17
18 #capitalize() and title()
19 text = "hello world"
20 capitalized_text = text.capitalize()
21 title_text = text.title()
22 print(capitalized_text)
23 print(title_text)
24
25 #strip()
26 text = "  Python  "
27 stripped_text = text.strip()
28 print(stripped_text)
29
30 #replace()
31 text = "Hello, World!"
32 new_text = text.replace(_old_="World", _new_="Python")
33 print(new_text)
```

Result



```
Run string Function x
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\string Function.py"
14
42
HELLO, PYTHON!
hello, python!
Hello world
Hello World
Python
Hello, Python!
['Hello,', 'how', 'are', 'you?']
Hello, how are you?
Process finished with exit code 0
```


Number Function

The image shows a PyCharm IDE window with the file 'Number function.py' open. The code in the editor is as follows:

```
1 #abs()
2
3 x = -5
4 absolute_value = abs(x)
5 print(absolute_value)
6
7 #round()
8
9 y = 4.8
10 rounded_value = round(y)
11 print(rounded_value)
12
13 #max() and min()
14
15 numbers = [3, 7, 1, 9, 4]
16 max_value = max(numbers)
17 min_value = min(numbers)
18 print(max_value)
19 print(min_value)
```

The Run console shows the command: `"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\Number function.py"`. The output is:

```
5
5
9
1
```

The process finished with exit code 0. The status bar at the bottom indicates the file is 'Number function.py' with settings: 2:1 CRLF UTF-8 4 spaces Python 3.12 (Data Engineering).

Python Function

The image shows a PyCharm IDE window with the file 'Function.py' open. The code in the editor is as follows:

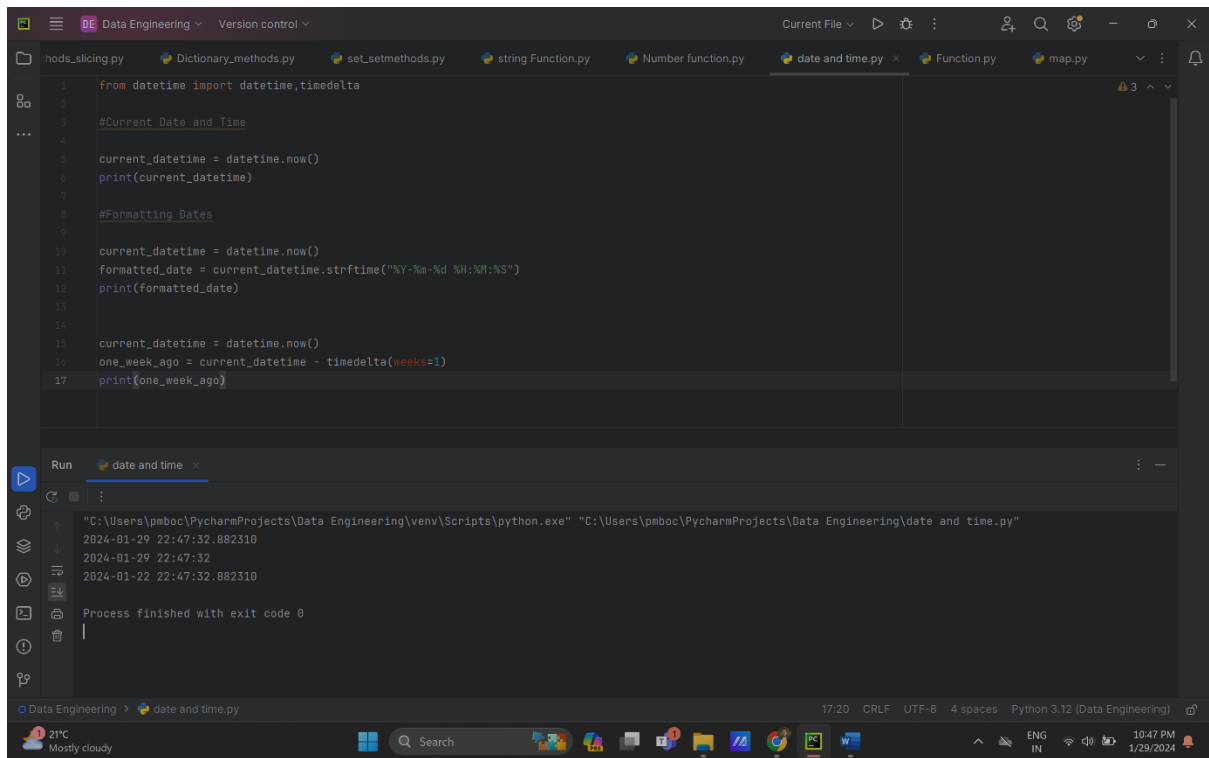
```
1
2 #factorial
3 2 usages
4 def factorial(n):
5
6     if n == 0 or n == 1:
7         return 1
8     else:
9         return n * factorial(n - 1)
10
11 result = factorial(5)
12 print(result)
13
14 #lambda function
15 multiply = lambda x, y: x * y
16 result = multiply(3, 4)
17 print(result)
18
```

The Run console shows the command: `"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\Function.py"`. The output is:

```
120
12
```

The process finished with exit code 0. The status bar at the bottom indicates the file is 'Function.py' with settings: 18:1 CRLF UTF-8 4 spaces Python 3.12 (Data Engineering).

Date and Time Function



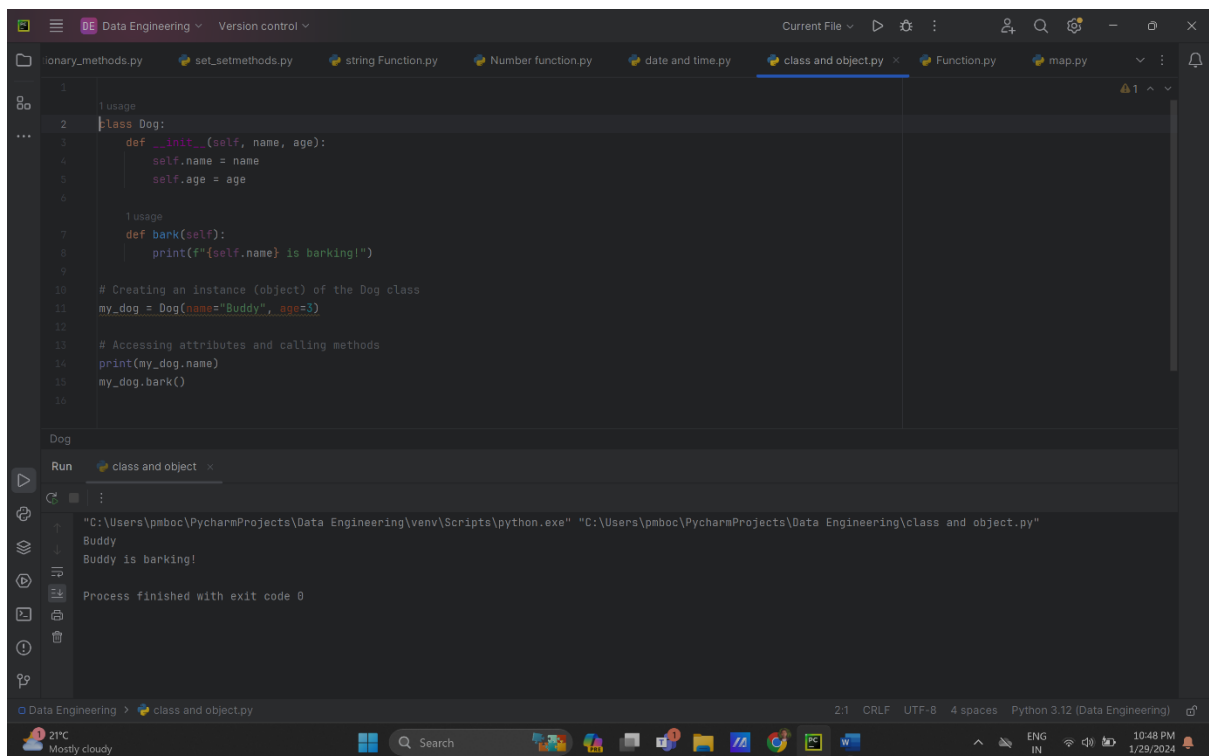
The screenshot shows a PyCharm IDE window titled "Data Engineering". The editor displays a Python script named "date and time.py". The script uses the `datetime` module to get the current date and time, format it, and calculate the time one week ago. The Run console shows the output of the script, which prints the current date and time, the formatted date, and the date one week ago. The status bar at the bottom indicates the file is encoded in UTF-8 with 4 spaces and is using Python 3.12.

```
1 from datetime import datetime, timedelta
2
3 #Current Date and Time
4
5 current_datetime = datetime.now()
6 print(current_datetime)
7
8 #Formatting Dates
9
10 current_datetime = datetime.now()
11 formatted_date = current_datetime.strftime("%Y-%m-%d %H:%M:%S")
12 print(formatted_date)
13
14
15 current_datetime = datetime.now()
16 one_week_ago = current_datetime - timedelta(weeks=1)
17 print(one_week_ago)
```

Run console output:

```
"C:\Users\pmboc\PycharmProjects\Data Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\date and time.py"
2024-01-29 22:47:32.882310
2024-01-29 22:47:32
2024-01-22 22:47:32.882310
Process finished with exit code 0
```

Class and Object



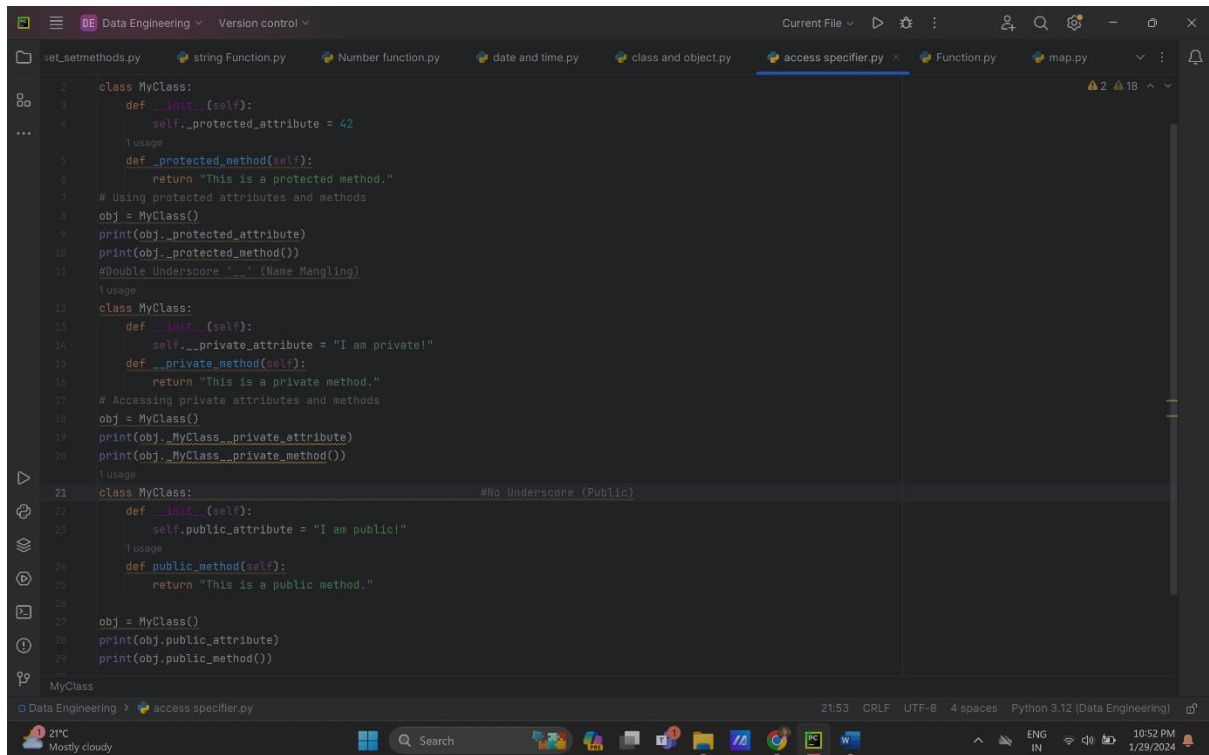
The screenshot shows a PyCharm IDE window titled "Data Engineering". The editor displays a Python script named "class and object.py". The script defines a `Dog` class with an `__init__` method and a `bark` method. It then creates an instance of the `Dog` class named `my_dog` and prints its name and barks. The Run console shows the output of the script, which prints the name of the dog and the message "Buddy is barking!". The status bar at the bottom indicates the file is encoded in UTF-8 with 4 spaces and is using Python 3.12.

```
1 #usage
2 class Dog:
3     def __init__(self, name, age):
4         self.name = name
5         self.age = age
6
7     #usage
8     def bark(self):
9         print(f"{self.name} is barking!")
10
11 # Creating an instance (object) of the Dog class
12 my_dog = Dog(name="Buddy", age=3)
13
14 # Accessing attributes and calling methods
15 print(my_dog.name)
16 my_dog.bark()
```

Run console output:

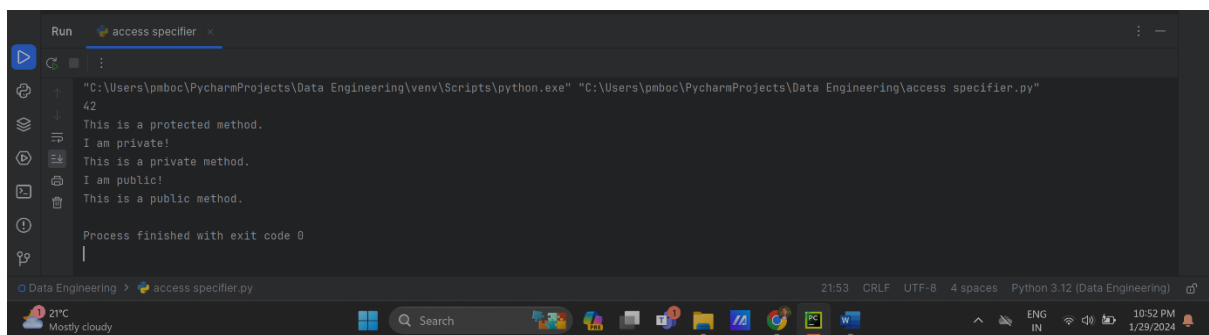
```
"C:\Users\pmboc\PycharmProjects\Data Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\class and object.py"
Buddy
Buddy is barking!
Process finished with exit code 0
```

Access Specifier



```
1 class MyClass:
2     def __init__(self):
3         self._protected_attribute = 42
4
5     def _protected_method(self):
6         return "This is a protected method."
7
8 # Using protected attributes and methods
9 obj = MyClass()
10 print(obj._protected_attribute)
11 print(obj._protected_method())
12 #Double Underscore "__" (Name Mangling)
13
14 class MyClass:
15     def __init__(self):
16         self.__private_attribute = "I am private!"
17     def __private_method(self):
18         return "This is a private method."
19
20 # Accessing private attributes and methods
21 obj = MyClass()
22 print(obj.__private_attribute)
23 print(obj.__private_method())
24
25 class MyClass:
26     #No Underscore (Public)
27     def __init__(self):
28         self.public_attribute = "I am public!"
29
30     def public_method(self):
31         return "This is a public method."
32
33 obj = MyClass()
34 print(obj.public_attribute)
35 print(obj.public_method())
```

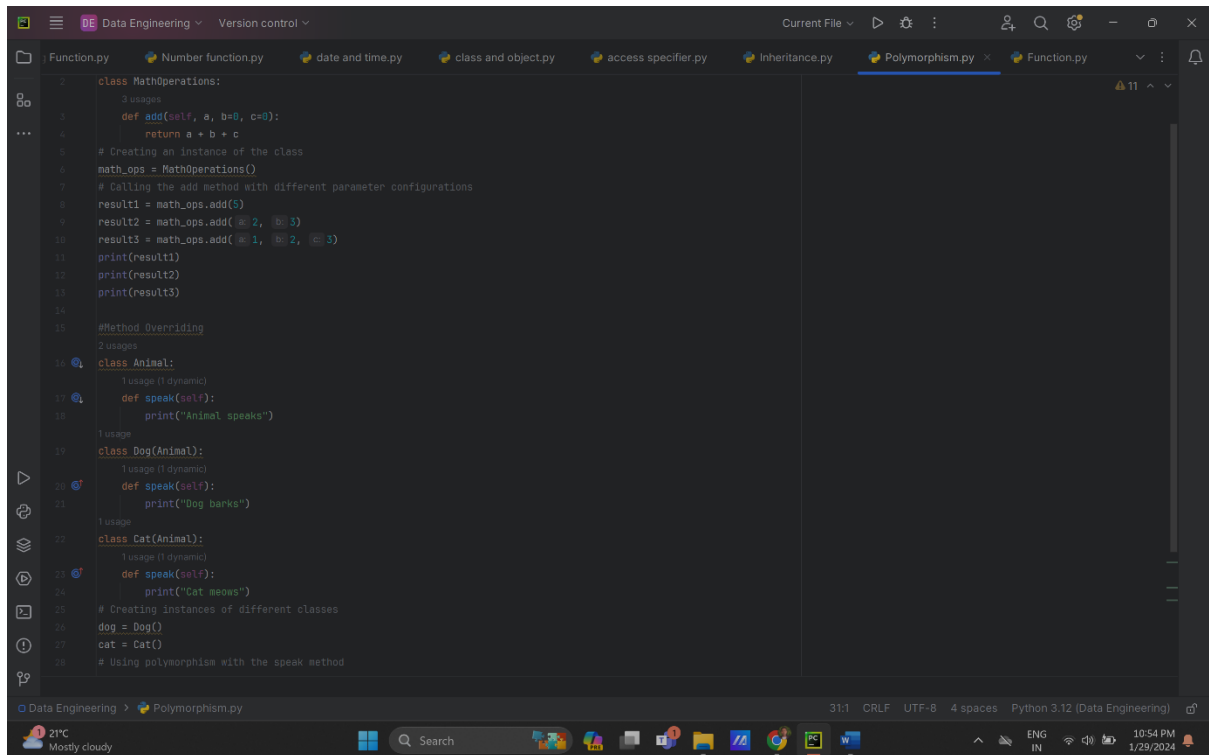
Result



```
Run access_specifier.py
42
This is a protected method.
I am private!
This is a private method.
I am public!
This is a public method.

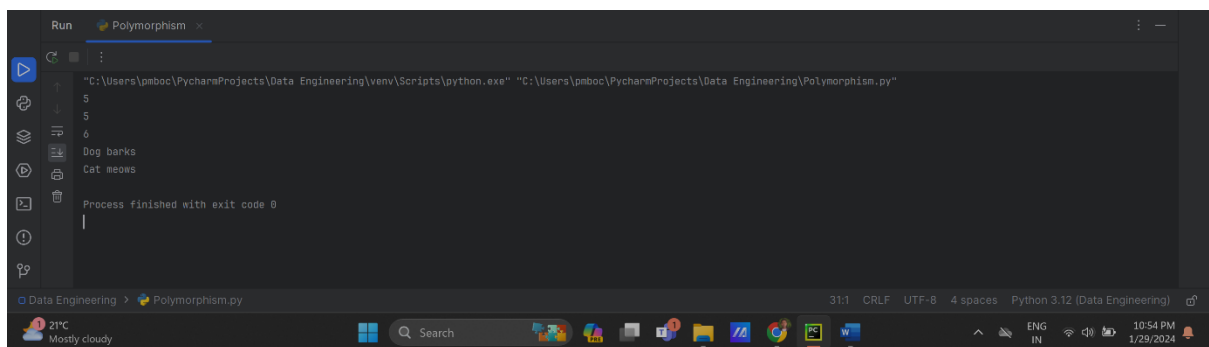
Process finished with exit code 0
```

Polymorphism



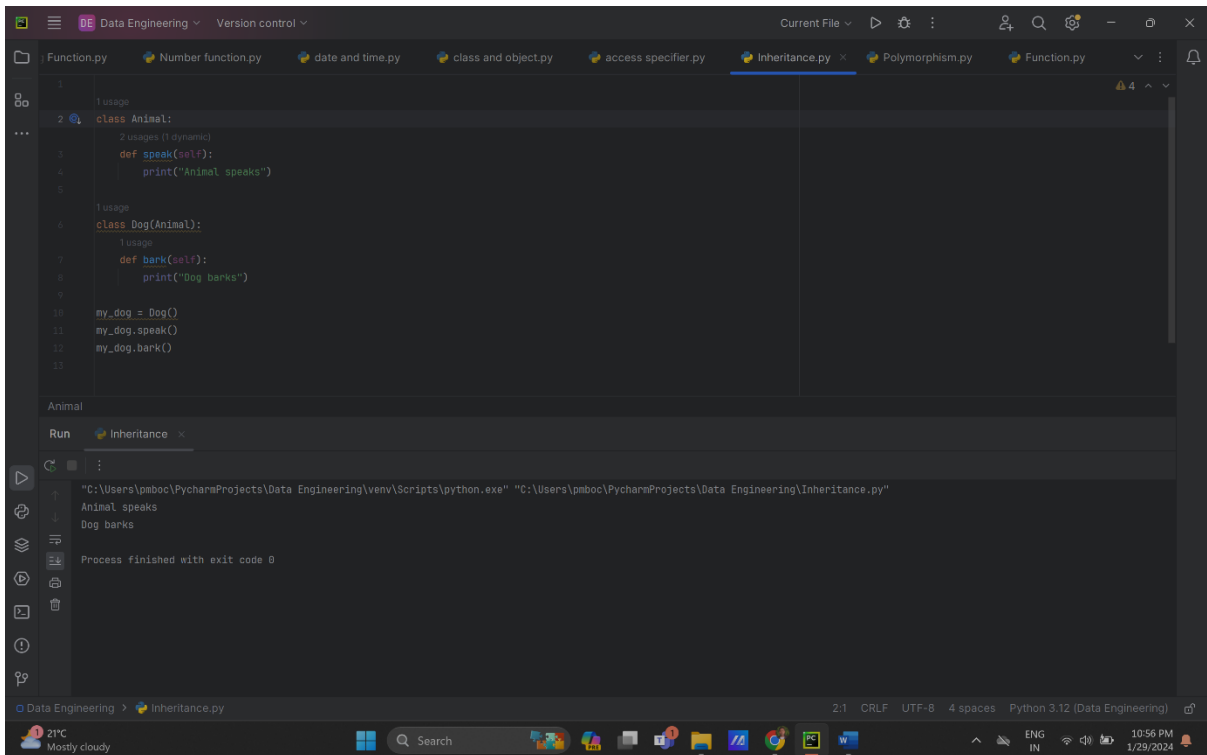
```
1 class MathOperations:
2     3 usages
3     def add(self, a, b=0, c=0):
4         return a + b + c
5
6     # Creating an instance of the class
7     math_ops = MathOperations()
8     # Calling the add method with different parameter configurations
9     result1 = math_ops.add(5)
10    result2 = math_ops.add(a=2, b=3)
11    result3 = math_ops.add(a=1, b=2, c=3)
12    print(result1)
13    print(result2)
14    print(result3)
15
16    #Method Overriding
17    2 usages
18    class Animal:
19        1 usage (1 dynamic)
20        def speak(self):
21            print("Animal speaks")
22
23        1 usage
24        class Dog(Animal):
25            1 usage (1 dynamic)
26            def speak(self):
27                print("Dog barks")
28
29        1 usage
30        class Cat(Animal):
31            1 usage (1 dynamic)
32            def speak(self):
33                print("Cat meows")
34
35        # Creating instances of different classes
36        dog = Dog()
37        cat = Cat()
38        # Using polymorphism with the speak method
```

Result



```
Run Polymorphism
C:\Users\pmboc\PycharmProjects\Data Engineering\venv\Scripts\python.exe "C:\Users\pmboc\PycharmProjects\Data Engineering\Polymorphism.py"
5
5
6
Dog barks
Cat meows
Process finished with exit code 0
```

Inheritance



The screenshot shows the PyCharm IDE with a project named "Data Engineering". The file "Inheritance.py" is open and contains the following Python code:

```
1
2 class Animal:
3     def speak(self):
4         print("Animal speaks")
5
6 class Dog(Animal):
7     def bark(self):
8         print("Dog barks")
9
10 my_dog = Dog()
11 my_dog.speak()
12 my_dog.bark()
```

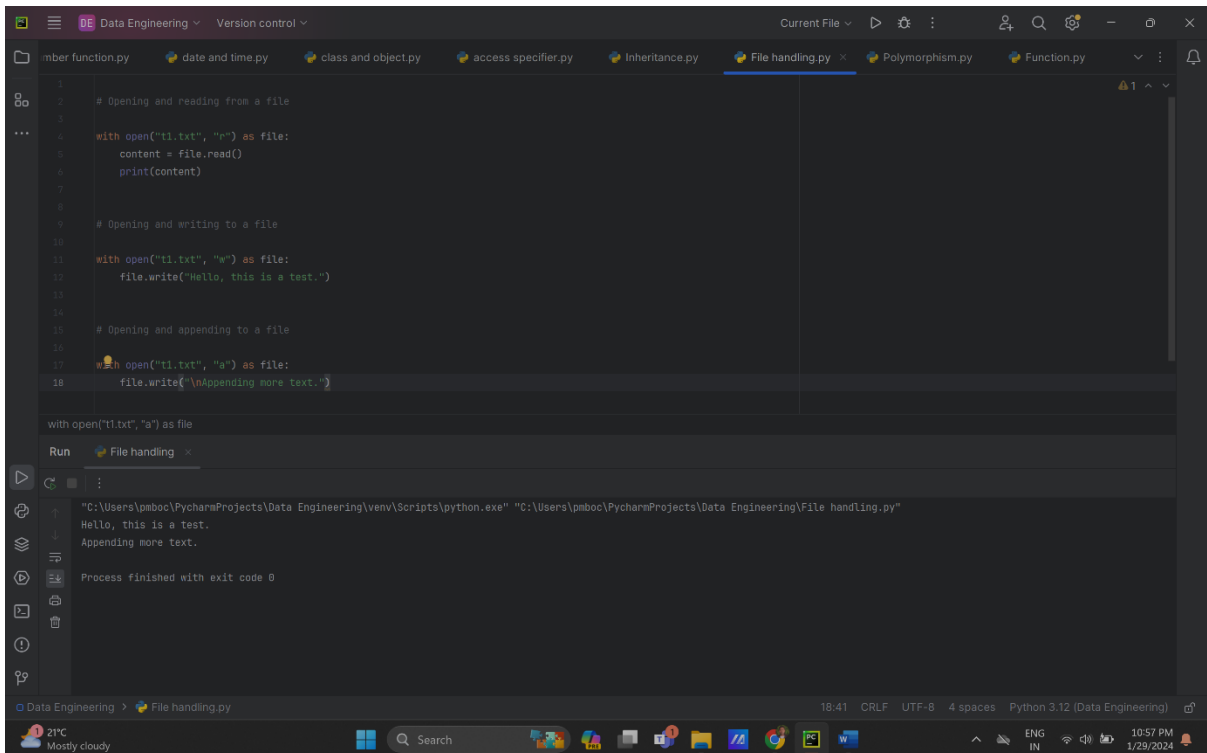
The code defines an `Animal` class with a `speak` method and a `Dog` class that inherits from `Animal` and has a `bark` method. An instance of `Dog` is created and both methods are called.

The Run console shows the output of the program:

```
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\Inheritance.py"
Animal speaks
Dog barks
Process finished with exit code 0
```

The status bar at the bottom indicates the file is "Inheritance.py" with settings: 2:1, CRLF, UTF-8, 4 spaces, Python 3.12 (Data Engineering).

File Handling



The screenshot shows the PyCharm IDE with a project named "Data Engineering". The file "File handling.py" is open and contains the following Python code:

```
1
2 # Opening and reading from a file
3
4 with open("t1.txt", "r") as file:
5     content = file.read()
6     print(content)
7
8
9 # Opening and writing to a file
10
11 with open("t1.txt", "w") as file:
12     file.write("Hello, this is a test.")
13
14
15 # Opening and appending to a file
16
17 with open("t1.txt", "a") as file:
18     file.write("\nAppending more text.")
```

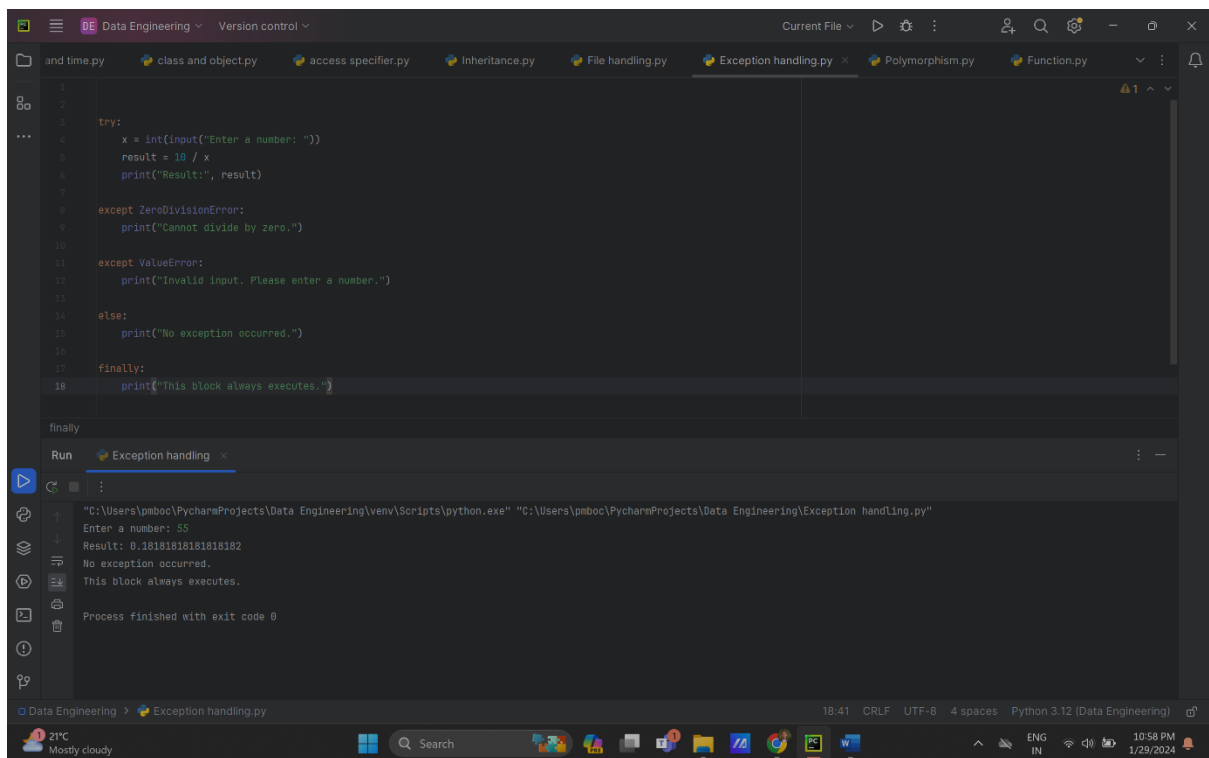
The code demonstrates three file operations: reading from a file, writing to a file, and appending to a file. Each operation is enclosed in a `with` block using the `open` function.

The Run console shows the output of the program:

```
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\File handling.py"
Hello, this is a test.
Appending more text.
Process finished with exit code 0
```

The status bar at the bottom indicates the file is "File handling.py" with settings: 18:41, CRLF, UTF-8, 4 spaces, Python 3.12 (Data Engineering).

Exception Handling



```
1
2
3 try:
4     x = int(input("Enter a number: "))
5     result = 10 / x
6     print("Result:", result)
7
8 except ZeroDivisionError:
9     print("Cannot divide by zero.")
10
11 except ValueError:
12     print("Invalid input. Please enter a number.")
13
14 else:
15     print("No exception occurred.")
16
17 finally:
18     print("This block always executes.")
19
20 finally:
```

Run Exception handling x

```
"C:\Users\pmboc\PycharmProjects\Data_Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data_Engineering\Exception_handling.py"
Enter a number: 55
Result: 0.18181818181818182
No exception occurred.
This block always executes.
Process finished with exit code 0
```

Data Engineering > Exception_handling.py 18:41 CRLF UTF-8 4 spaces Python 3.12 (Data Engineering)

21°C Mostly cloudy 10:58 PM 1/29/2024