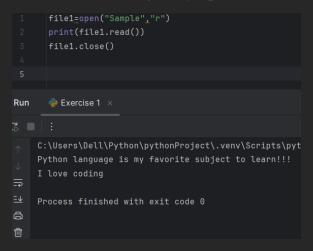
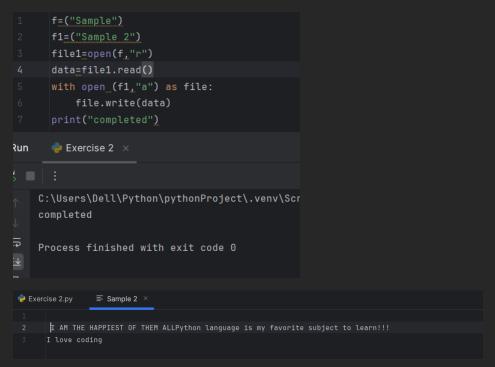
Exercise 1: Write a Python program to read a file and display its contents



Exercise 2: Write a Python program to copy the contents of one file to another file



Exercise 3:Write a Python program to read the content of a file and count the total number of words in that file.

```
file1=open("Sample","r")

count=0

for line in file1:
    words = line.split(" ")
    count+=len(words)

file1.close()

print("Number of words in a Sample textfile is",count)

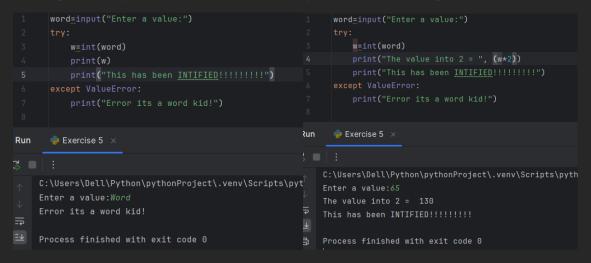
Run    Exercise 3 ×

C:\Users\Dell\Python\pythonProject\.venv\Scripts\python.exe '
Number of words in a Sample textfile is 11

Process finished with exit code 0
```

Exercise 4: Write a Python program to read the content of a file and count the number of occurrences of a specific word in that file.

Exercise 5: Write a Python program that prompts the user to input a string and converts it to an integer. Use try-except blocks to handle any exceptions that might occur



Exercise 6: Write a Python program that prompts the user to input a list of integers and raises an exception if any of the integers in the list are negative.

```
numbers = input("Enter a new list:")

print(type(numbers))

l1=list(map(int,numbers.split(",")))

print(l1)

print(type(l1))

l2=[]

try:

for num in l1:

if num≥=0:

l2.append(num)

print("List without negative value:",12)

except ValueError:

print("There is no negative values")

Run  Exercise 6 ×

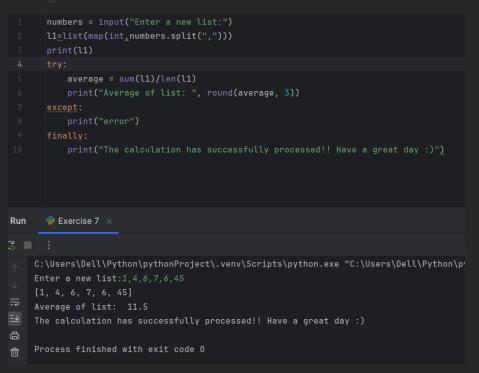
C:\Users\Dell\Python\pythonProject\.venv\Scripts\python.exe "C:\Use Enter a new list:1,5,1,4,6,-8,-9

<class 'str'>
[1, 5, 1, 4, 6, -8, -9]

<class 'list'>
List without negative value: [1, 5, 1, 4, 6]

Process finished with exit code 0
```

Exercise 7: Write a Python program that prompts the user to input a list of integers and computes the average of those integers. Use try-except blocks to handle any exceptions that might occur.use the finally clause to print a message indicating that the program has finished running.



Exercise 8: Write a Python program that prompts the user to input a filename and writes a string to that file. Use try-except blocks to handle any exceptions that might occur and print a welcome message if there is no exception occurred.

```
try:
    file_name = input("Enter file name: ")
    file=open(file_name,"w")
    file.write("\n I AM THE HAPPIEST OF THEM ALL")
    file.close()
    print("Hello, Welcome to Python")
    except FileNotFoundError:
    print('File you entered is not found')

un
    Exercise 8 ×

:
    C:\Users\Dell\Python\pythonProject\.venv\Scripts\python.exe "
    Enter file name: Sample4
    Hello, Welcome to Python

Process finished with exit code 0
```

Exercise 9 : Build a program to manage a university's course catalog. You want to define a base class Course that has the following properties:

course code: a string representing the course code (e.g., "CS101")

course_name: a string representing the course name (e.g., "Introduction to Computer Science")

credit_hours: an integer representing the credit hours for the course (e.g., 3)

You also want to define two subclasses CoreCourse and ElectiveCourse, which inherit from the Course class.

CoreCourse should have an additional property required_for_major which is a boolear representing whether the course is required for a particular major.

ElectiveCourse should have an additional property elective_type which is a string representing the type of elective (e.g., "general", "technical", "liberal arts").

```
class Course:

class Course
class Course
class Course
course_data(self_course_code_course_name_credit_hours):

def Course_data(self_course_code
self_course_code=course_code
self_course_name_course_name
self_credit_hours=credit_hours
print("Nourse Code :", course_code)
print("Course Name : ", course_name)
print("Credit Hours : ", credit_hours)

tusage
class CoreCourse(Course):

def __init__(self_course_code_course_name_credit_hours):
super().Course_data(course_code_course_name_credit_hours)

2 usages

def required(self_required_for_major):

if required_for_major=i:
    print("\nRequired for major")

else:
    print("\nRequired for major")

tusage

class ElectiveCourse(Course):

def __init__(self_course_code, course_name, credit_hours,elective_type):

super().Course_data(course_code, course_name, credit_hours)

self_elective_type=elective_type
print("Elective_type=elective_type)
```

```
code=input("Enter the course code:")
name = input("Enter the course name:")
hours = input("Enter the course time:")
try:
    required=input('\nEnter whether the course is required or not (Yes/No) :')
        corcourse=CoreCourse(code,name_hours)
    if required.lower()=="yes":
        corcourse.required(1)
    elif required.lower()=="no":
        corcourse.required(0)
    else:
        print("\n Enter a valid input.....PLEASE!!")

    elective_type=input(("\nEnter elective subject:"))
    elective_c=ElectiveCourse(code_name_hours_elective_type)
except Exception as error:
    print("An error occured please check",error)
```

```
C:\Users\Dell\Python\pythonProject\.venv\Scripts\python.ex
Enter the course code:CS101
Enter the course name:Introduction to Computer Science
Enter the course time:3

Enter whether the course is required or not (Yes/No) :Yes

Course Code : CS101
Course Name : Introduction to Computer Science
Credit Hours : 3

Required for major

Enter elective subject:Liberal Arts

Course Code : CS101
Course Name : Introduction to Computer Science
Credit Hours : 3

Elective subject is : Liberal Arts

Process finished with exit code 0
```

Exercise 10: Create a Python module named employee that contains a class Employee with attributes name, salary and methods get_name() and get_salary(). Write a program to use this module to create an object of the Employee class and display its name and salary.

```
4 usages

class Employee:

emptotal = 0

def __init__(self,name,salary):

self.name=name

self.salary=salary

Employee.emptotal += 1

lusage

def get_count(self):

print("Total employees:",self.get_count())

3 usages

def get_name(self):

print("Employee Name:",self.name)

3 usages

def get_salary(self):

print(" Salary:",self.salary)
```

```
import Employee_info
E1=Employee_info.Employee( name: "Mathew", salary: 35000)
E2=Employee_info.Employee( name: "Steffy", salary: 40000)
E3=Employee_info.Employee( name: "Stanley", salary: 50000)
print("Employee details:")
print(E1.get_name(),"\t",E1.get_salary())
print(E2.get_name(),"\t",E2.get_salary())
print(E3.get_name(),"\t",E3.get_salary())

C:\Users\Dell\Python\pythonProject\.venv\Scr
Employee details:
Employee Name: Mathew
Salary: 35000
None None
```

Employee Name: Steffy Salary: 40000

Employee Name: Stanley

Process finished with exit code 0

None

None