ASSIGNMENT-4

1. What is File function in python? What is keywords to create and write file.

➢ In Python, the **open()** function is commonly used to work with files. The **open()** function returns a file object that allows you to interact with the file, such as reading from it or writing to it

Here are some key parameters for the **open()** function:

- **'w'**: Opens the file for writing. If the file doesn't exist, it creates a new file. If the file exists, it truncates the file (removes the existing content).
- **'a'**: Opens the file for appending. If the file doesn't exist, it creates a new file. If the file exists, it appends new content to the end of the file.
- **'r'**: Opens the file for reading.

2. Explain Exception handling? What is an Error in Python?

➢ Exception handling is a mechanism in Python that allows you to deal with errors and exceptions that may occur during the execution of a program. Errors can happen for various reasons, such as invalid input, file not found, division by zero, etc. Exception handling helps you gracefully handle these situations, preventing your program from crashing and providing a way to respond to errors

➢ **ERROR:**

➢ An error is a problem that occurs at runtime, causing the program to terminate.

➢ Errors are typically categorized as syntax errors or runtime errors.

➢ Syntax errors occur during the parsing of code, and the program won't run if there are syntax errors.

➢ Runtime errors occur during the execution of the program and may cause it to terminate unexpectedly.

3. How many except statements can a try-except block have? Name Some built-in exception classes:

➢ A **try-except** block in Python can have multiple **except** statements to handle different types of exceptions. Each **except** block corresponds to a specific exception class or a tuple of exception classes. You can have as many **except** blocks as needed to handle different types of exceptions. The first **except** block that matches the raised exception will be executed, and subsequent **except** blocks will be skipped.

➢ Some common built-in exception classes in Python include:

➢ **ZeroDivisionError**: Raised when division or modulo operation is performed with zero as the divisor.

➢ **ValueError**: Raised when a built-in operation or function receives an argument of the correct type but an invalid value.

➢ **TypeError**: Raised when an operation or function is applied to an object of an inappropriate type.

➢ **FileNotFoundError**: Raised when a file or directory is requested but cannot be found.

➢ **IndexError**: Raised when a sequence subscript is out of range.

➢ **KeyError**: Raised when a dictionary key is not found.

➢ **NameError**: Raised when a local or global name is not found.

- ➢ **IOError**: Raised when an I/O operation (such as reading or writing to a file) fails.
- ➢ **SyntaxError**: Raised when the parser encounters a syntax error.

4. When will the else part of try-except-else be executed?

➢ In a **try-except-else** block in Python, the **else** block is executed only if the code in the **try** block runs successfully without raising any exceptions. If any exception occurs within the **try** block, the **except** block is executed, and the **else** block is skipped.

```
try:
# Code that might raise an exception
# ...
except SomeException:
    # Code to handle SomeException
    # ...
else:
    # Code to be executed if no exception occurred in the try block
```

5. Can one block of except statements handle multiple exception?

➢ Yes, a single **except** block in Python can handle multiple exceptions by specifying them as a tuple. This allows you to catch and handle different types of exceptions with the same block of code. Here's an example:

6. When is the finally block executed?

➢ The **finally** block in a Python **try-except-finally** or **try-finally** construct is executed no matter what, whether an exception is raised or not. The primary purpose of the **finally** block is to define cleanup actions that should be performed, such as closing files or releasing resources, regardless of whether an exception occurs.

7. What happens when „1‟== 1 is executed?

➢ When the expression **"1" == 1** is executed in Python, it will evaluate to **False**. This is because the equality operator (**==**) in Python compares the values on both sides for equality, and in this case, the types of the values are different.

➢ The left side of the expression is a string literal **"1"**, which is a sequence of characters, and the right side is the integer literal **1**. Although the characters in the string represent the digit 1, the string and the integer are of different types.

➢ In Python, when comparing values of different types using the equality operator, the result is always **False**. If you want to compare the values regardless of their types, you might need to convert one of the values to the type of the other.

8. How to Define a Class in Python? What Is Self? Give An Example Of A Python Class

- In Python, you can define a class using the **class** keyword. A class is a blueprint for creating objects, which are instances of that class. Objects have attributes (variables) and methods (functions) associated with them. The **self** keyword is used to refer to the instance of the class itself. It is the first parameter in the definition of a method in a class and is a convention in Python.

9. What is Instantiation in terms of OOP terminology?

- In object-oriented programming (OOP), instantiation refers to the process of creating an instance of a class, which is an individual object of that class type. A class is essentially a blueprint or template that defines the structure and behavior of objects. When you create an instance of a class, you are creating a specific object that follows the blueprint specified by the class.

10. What is used to check whether an object o is an instance of class A?

- In Python, you can use the **isinstance()** function to check whether an object **o** is an instance of a particular class, such as class **A**. The **isinstance()** function takes two arguments: the object you want to check, and the class (or a tuple of classes) you want to check against.

11. What relationship is appropriate for Course and Faculty?

- The relationship between a "Course" and "Faculty" in a university or educational context is typically represented as an association, and the appropriate type of association depends on the nature of the relationship.

12. What relationship is appropriate for Student and Person?

- In the context of a university or educational environment, the relationship between "Student" and "Person" is typically an inheritance relationship, indicating that a "Student" is a specialized type of "Person." This relationship is expressed using the concept of class inheritance, where the "Student" class is a subclass of the "Person" class.