Python Model Paper -1

1. **What is an arithmetic expression? What is the output of this statement? ‘hello world’ + 100 + ’how are you’ explain the reason if the statement produces an error.**

   An arithmetic expression is a combination of variables, constants, and mathematical operations such as addition, subtraction, multiplication, and division that can be evaluated to a numerical value.

   The above statement produces an type error as python does not allow for concatenation of different data types such as integer and a string value as shown above.

2. **Discuss various methods of importing modules in Python programs. Which method is best? Explain.**

➢

➢ `import module_name`: This is the simplest method of importing a module. It allows access to all the functions and variables defined in the module using the `module_name.function_name` syntax.

2. `from module_name import function_name`: This method allows importing specific functions or variables from a module, making them available in the main program without using the module prefix.
3. `import module_name as alias`: This method allows importing a module using an alias, which can make the code more readable by providing a shorter or more descriptive name.
4. `from module_name import *`: This method imports all functions and variables from the specified module into the current namespace. However,

it is generally not recommended because it can lead to naming conflicts and make the code harder to read and debug.

The best method of importing modules in Python depends on the specific needs of the program and personal preference. However, it is generally recommended to use the first or second method, which provides clear and explicit references to the imported functions or variables.

3. **What is the lambda function? Explain with an example of addition of two numbers.**

A lambda function is a small, anonymous function that can be defined on a single line without a name using the `lambda` keyword. Lambda functions are often used as a convenient way to define simple functions on the fly, without the need to define a separate named function.

Here is an example of a lambda function that adds two numbers:

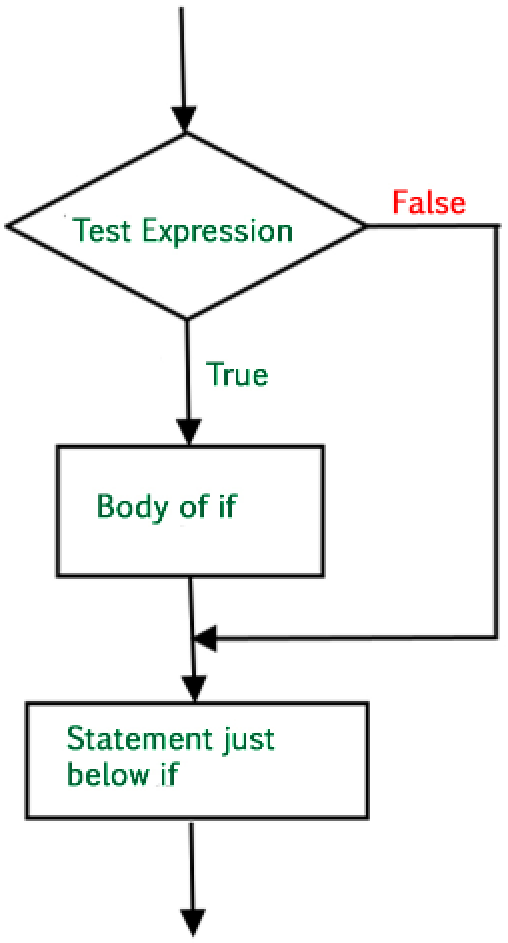
```
addition = lambda x, y: x + y

result = addition(2, 3)

print(result)  # Output: 5
```

In this example, the lambda function is defined with the `lambda` keyword, followed by the arguments `x` and `y`, separated by a comma. The body of the lambda function follows the `:` symbol, which performs the addition of the two arguments and returns the result.

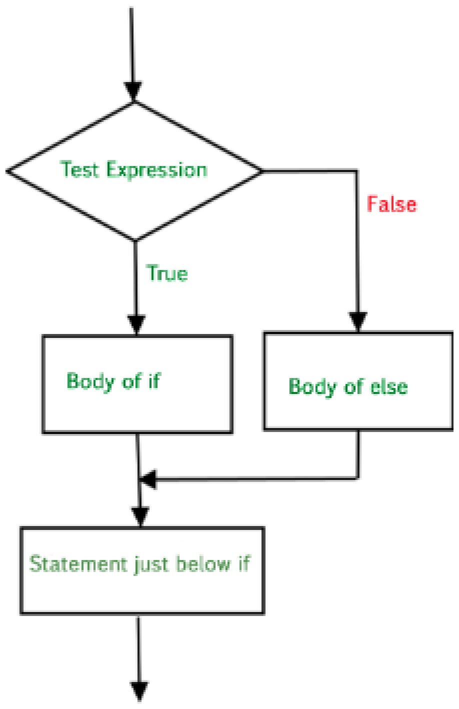
4. **What is a flow control statement?. Discuss if and if else statements with flow chart.**

   In programming, a flow control statement is a command that directs the flow of execution of a program. Flow control statements enable a program to make decisions based on certain conditions, execute code repeatedly, or jump to different parts of the program based on specific criteria.

   The `if` statement is used to execute a block of code if a certain condition is true.

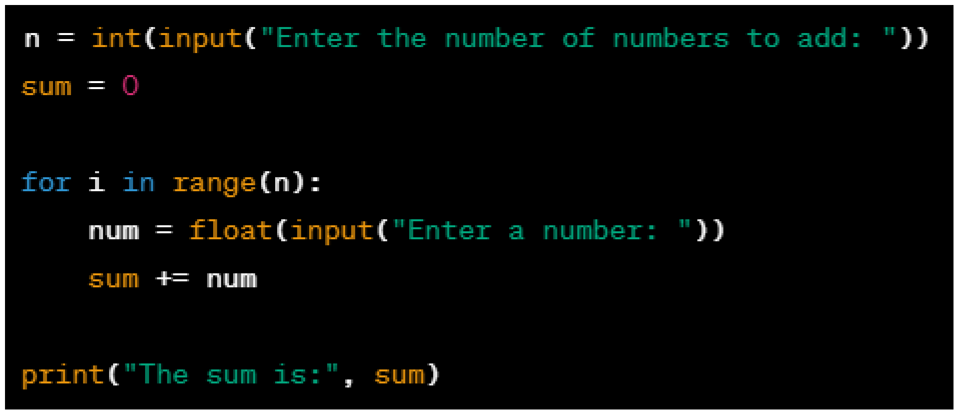


The `if-else` statement is used to execute one block of code if a certain condition is true and another block of code if the condition is false.



NOTE:- Add rectangular End box in both the flow chart.

5. **Write a python program to add n numbers accepted from the user.**



```
n = int(input("Enter the number of numbers to add: "))
sum = 0

for i in range(n):
    num = float(input("Enter a number: "))
    sum += num

print("The sum is:", sum)
```

6. **How can you prevent a python program from crashing? discuss different ways to avoid crashing.**

1. Exception handling

Python provides a built-in mechanism for handling exceptions that can occur during program execution. By using `try` and `except` statements, you can catch and handle exceptions that might cause your program to crash.

2. Defensive programming

Defensive programming involves anticipating and handling potential errors or bugs in your code. One way to do this is to use conditional statements to check for edge cases or unexpected input. For example, you might check that a variable is not `None` before using it, or that a list has a certain length before accessing an index. By checking for potential errors ahead of time, you can prevent your program from crashing due to unexpected behavior.

3. Debugging and testing

Debugging and testing are critical parts of software development that can help prevent crashes and errors. By testing your code thoroughly and using debugging tools to identify and fix bugs, you can catch potential issues before they cause your program to crash.

4. Robust input validation

Input validation is the process of checking that user input is correct and conforms to expected formats. By robustly validating input, you can prevent your program from crashing due to unexpected or malicious user input. For example, if your program expects a number as input, you might check that the input is a valid number and not a string or other type of data.

7. **Discuss list and dictionary data structure with example for each.**

A list is an ordered collection of values, which can be of any data type. You can add, remove, and modify elements of a list, and you can access individual elements by their position in the list (also called an index). Here's an example:

```
# create a list of integers

my_list = [1, 2, 3, 4, 5]

# add an element to the end of the list

my_list.append(6)

# remove an element from the list

my_list.remove(3)

# access an element by index

print(my_list[2])  # prints 4
```

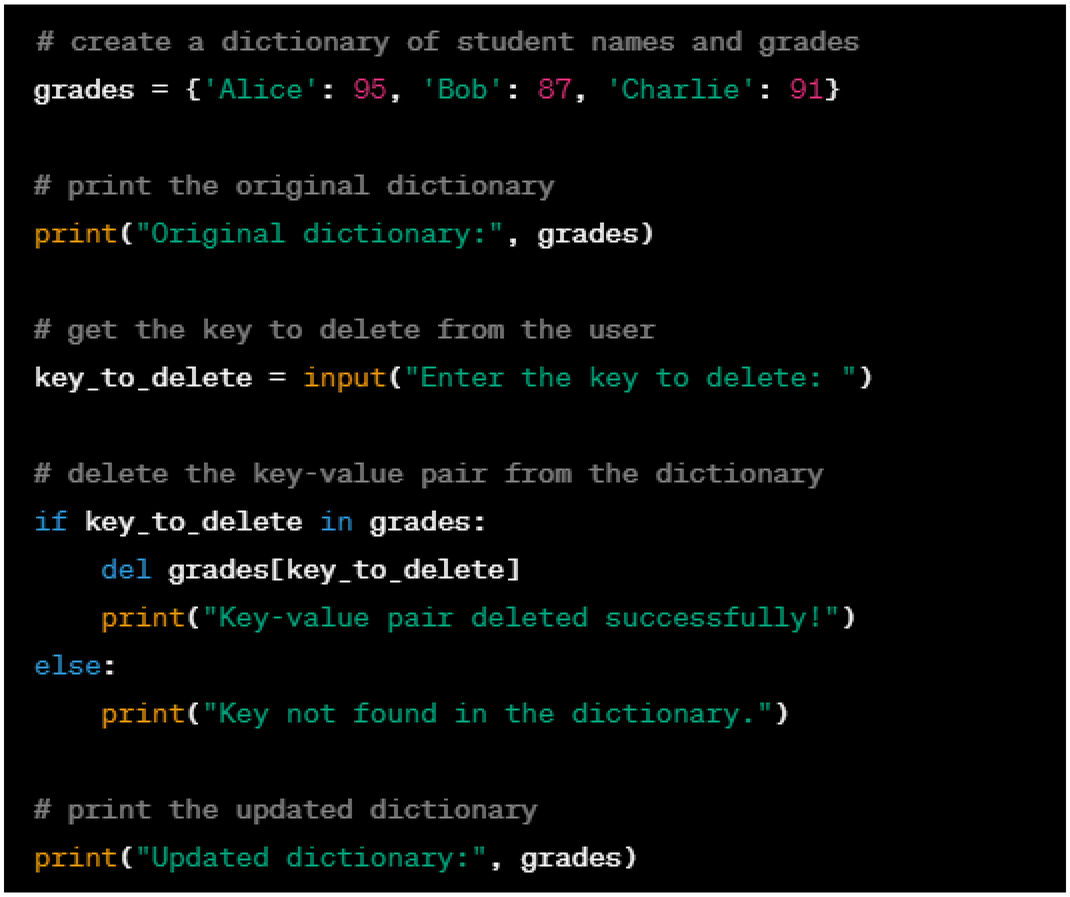
A dictionary is an unordered collection of key-value pairs, where each key is associated with a value. You can add, remove, and modify key-value pairs in a dictionary, and you can access individual values by their corresponding key. Here's an example:

```
# create a dictionary of student names and grades
grades = {'Alice': 95, 'Bob': 87, 'Charlie': 91}

# add a new key-value pair to the dictionary
grades['David'] = 79

# remove a key-value pair from the dictionary
del grades['Bob']

# access a value by key
print(grades['Charlie'])  # prints 91
```

8. **write a Python program to accept n numbers and store them in a list. Then print the list without ODD numbers in it.**

```
n = int(input("Enter the number of elements in the list: "))
my_list = []

# accept n numbers from the user and store them in the list
for i in range(n):
    num = int(input("Enter element {}:".format(i+1)))
    my_list.append(num)

# create a new list with only the even numbers from the original list
new_list = [num for num in my_list if num % 2 == 0]

# print the new list without any odd numbers
print("The list without odd numbers is: ", new_list)
```

9. **write a python program to read dictionary data and delete any given key entry in the dictionary.**



```python
# create a dictionary of student names and grades
grades = {'Alice': 95, 'Bob': 87, 'Charlie': 91}

# print the original dictionary
print("Original dictionary:", grades)

# get the key to delete from the user
key_to_delete = input("Enter the key to delete: ")

# delete the key-value pair from the dictionary
if key_to_delete in grades:
    del grades[key_to_delete]
    print("Key-value pair deleted successfully!")
else:
    print("Key not found in the dictionary.")

# print the updated dictionary
print("Updated dictionary:", grades)
```

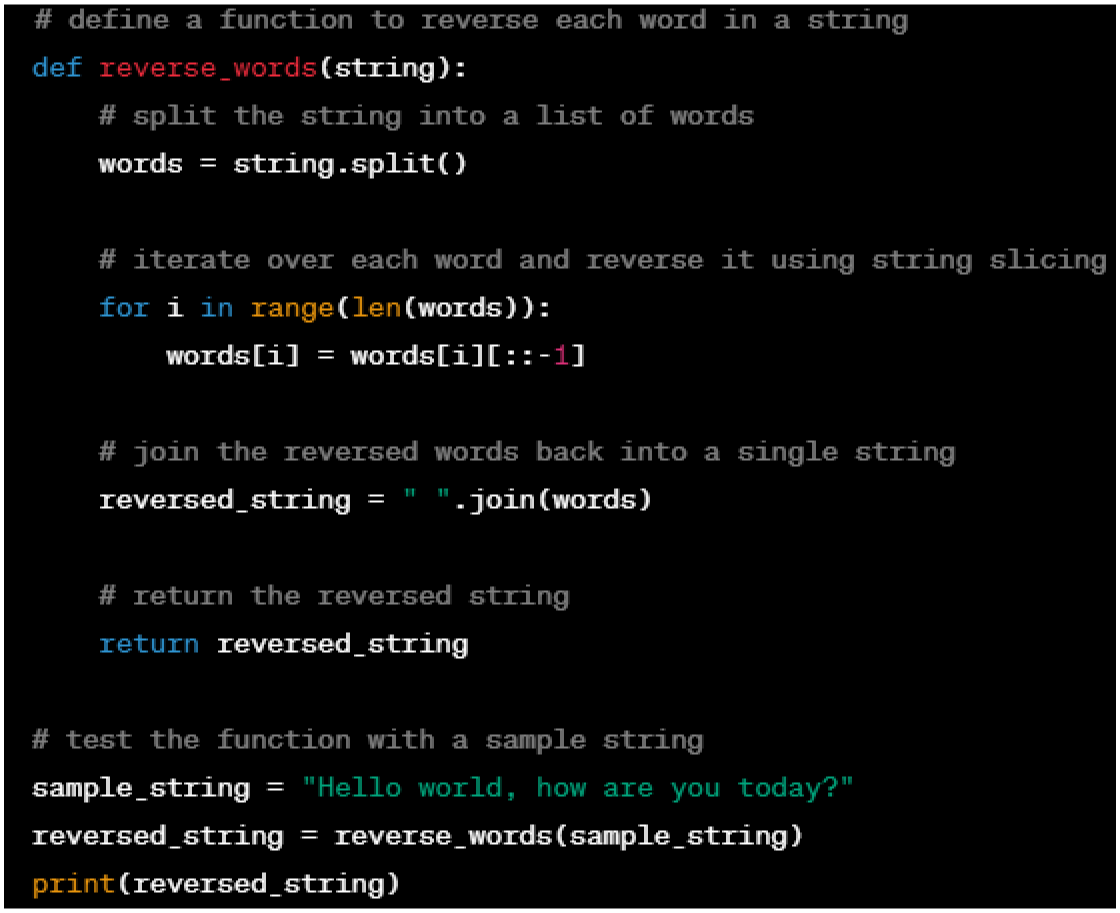
10. **Explain different clipboard functions in python used in wiki markup.**

Some of the commonly used clipboard functions in Python for wiki markup are:

➢ `pyperclip.copy(text)` - This function copies the text argument to the clipboard. For example, `pyperclip.copy('hello world')` will copy the string "hello world" to the clipboard.

- `pyperclip.paste()` - This function returns the text currently on the clipboard. For example, `my_text = pyperclip.paste()` will retrieve the text currently on the clipboard and store it in the variable `my_text`.
- `pyperclip.waitForNewPaste()` - This function waits for new text to be copied to the clipboard and returns it. This can be useful in situations where the program needs to wait for the user to copy text to the clipboard before continuing.

11. Using string slicing operation write a Python program to reverse each word in a given string



```python
# define a function to reverse each word in a string
def reverse_words(string):
    # split the string into a list of words
    words = string.split()

    # iterate over each word and reverse it using string slicing
    for i in range(len(words)):
        words[i] = words[i][::-1]

    # join the reversed words back into a single string
    reversed_string = " ".join(words)

    # return the reversed string
    return reversed_string

# test the function with a sample string
sample_string = "Hello world, how are you today?"
reversed_string = reverse_words(sample_string)
print(reversed_string)
```

NOTE:- In the above program use input() function for sample_string variable to get the string from the user instead of the predetermined string.

12. Discuss different paths of file system.

➢

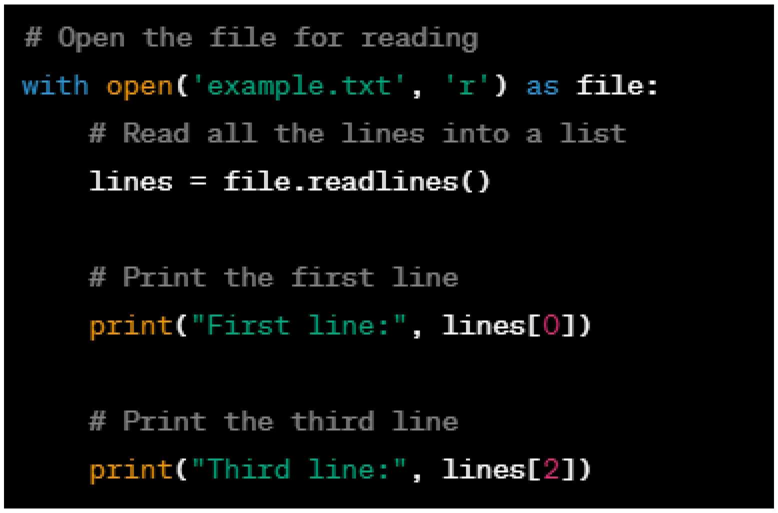
➢ Absolute path: An absolute path specifies the full path to a file or directory from the root of the file system. It always starts with a forward slash ("/") on Unix-based systems or a drive letter followed by a colon (":") on Windows systems. For example, `/home/user/Documents/myfile.txt` is an absolute path to a file on a Unix-based system.

➢ Relative path: A relative path specifies the path to a file or directory relative to the current working directory. It does not start with a forward slash or a drive letter. For example, if the current working directory is `/home/user/Documents`, then `myfile.txt` is a relative path to a file in that directory.

➢ Home path: A home path allows you to reference the home directory of the current user. It uses the tilde ("~") character followed by the username to specify the home directory. For example, `~/Documents` is a home path to the Documents directory of the current user.

13. Explain how to read specific lines from a file?. illustrate with python program.

To read specific lines from a file in Python, you can use the `readlines()` method to read all the lines into a list, and then access the specific lines you need using list indexing. Here's an example program that demonstrates how to do this:



```python
# Open the file for reading
with open('example.txt', 'r') as file:
    # Read all the lines into a list
    lines = file.readlines()

    # Print the first line
    print("First line:", lines[0])

    # Print the third line
    print("Third line:", lines[2])
```

14. What is logging? how this would be used to debug the python program?

Logging is a built-in module in Python that allows you to record messages about the execution of your program. It provides a way to capture information about what your program is doing and store it in a file or display it on the console. Logging is a very useful tool for debugging Python programs because it can help you identify and diagnose issues in your code.

To use logging to debug a Python program, you would typically follow these steps:

➢ Import the logging module using the `import logging` statement.
➢ Configure the logging system using the `basicConfig()` method. This method sets the format of the log messages and specifies where the log messages should be sent (for example, to a file or to the console).
➢ Use logging statements to record information about the execution of your program. There are several levels of logging statements available, from

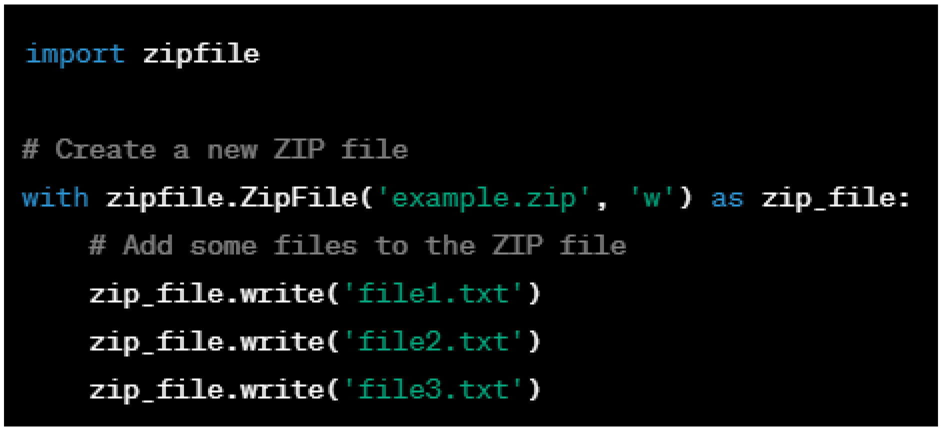
`debug` (for detailed information about the program's execution) to `error` (for reporting errors in the program).

➢ Analyze the log messages to identify and diagnose issues in your code.

15. What is the use of ZIP? how to create a ZIP folder explain.

ZIP is a popular file compression and archival format that is widely used for packaging and distributing files. It is a way to compress multiple files into a single archive file, making it easier to share or transfer large sets of files. ZIP files can be created and opened on almost any operating system, including Windows, macOS, and Linux.

In Python, the `zipfile` module provides functionality for working with ZIP files. You can use this module to create a new ZIP file and add files to it, or to extract files from an existing ZIP file.



```python
import zipfile

# Create a new ZIP file
with zipfile.ZipFile('example.zip', 'w') as zip_file:
    # Add some files to the ZIP file
    zip_file.write('file1.txt')
    zip_file.write('file2.txt')
    zip_file.write('file3.txt')
```

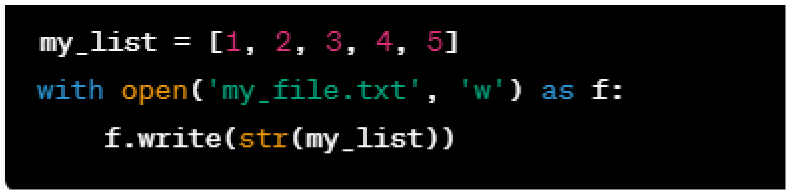
16. write an algorithm for implement multi clipboard functionality

Here's an algorithm for implementing a multi-clipboard functionality:

- ➢ Create an empty dictionary called `clipboard` to store the copied text and a variable called `count` to keep track of the number of items in the clipboard.
- ➢ Implement a function called `copy()` that takes a string as input and adds it to the clipboard with a unique key generated by incrementing the `count` variable.
- ➢ Implement a function called `paste()` that takes an integer key as input and returns the string associated with that key from the clipboard.
- ➢ Implement a function called `list_clipboard()` that prints all the keys and their associated strings in the clipboard.
- ➢ Implement a loop that repeatedly prompts the user for input and performs the appropriate operation based on the input. The loop should continue until the user enters a specific command to exit the program.

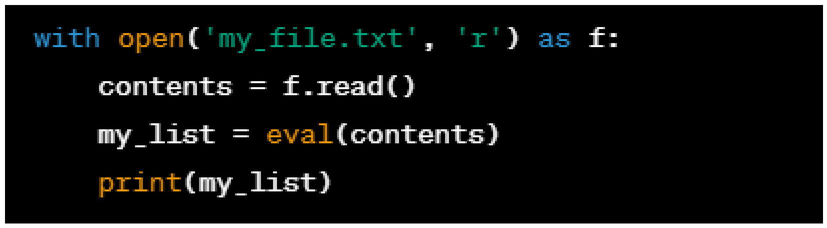
17. Discuss how lists would be written in the file and read from the file?

To write a list to a file in Python



```python
my_list = [1, 2, 3, 4, 5]
with open('my_file.txt', 'w') as f:
    f.write(str(my_list))
```

To read a list from a file in Python

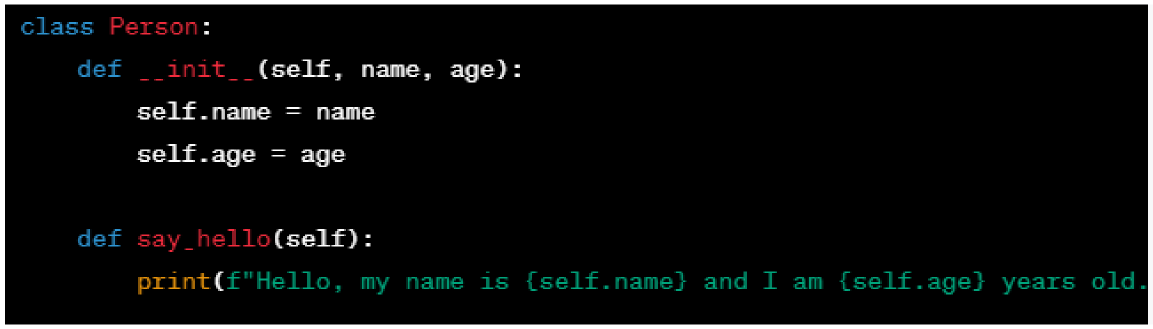


```python
with open('my_file.txt', 'r') as f:
    contents = f.read()
    my_list = eval(contents)
    print(my_list)
```

NOTE:~ Explain the above code in detail

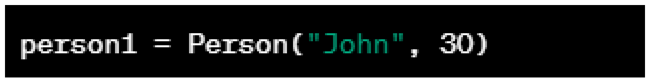
18. Define the terms with example: (i) class (ii) objects (iii) instance variables.

(i) Class: A class is a blueprint or a template for creating objects that defines a set of attributes and methods that the objects will have. In Python, you can define a class using the `class` keyword followed by the name of the class. Here's an example:



```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def say_hello(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.
```

(ii) Object: An object is an instance of a class. It is created from the blueprint or template provided by the class and has its own set of attributes and methods. In Python, you can create an object by calling the class as if it were a function. Here's an example:



```python
person1 = Person("John", 30)
```

In this example, we create a `Person` object called `person1` with a `name` attribute of "John" and an `age` attribute of 30.

(iii) Instance variables: Instance variables are variables that belong to a specific instance of a class, as opposed to class variables, which are shared among all instances of a class. In Python, you can define instance variables by assigning values to them within the `__init__()` method of the class. Here's an example:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.height = None

    def say_hello(self):
        print(f"Hello, my name is {self.name}, I am {self.age} years old, ar
```

19. create a Time class with hour, min and sec as attributes. Demonstrate how two Time objects would be added.

```
class Time:
    def __init__(self, hour=0, minute=0, second=0):
        self.hour = hour
        self.minute = minute
        self.second = second

    def __add__(self, other):
        total_seconds = self.to_seconds() + other.to_seconds()
        return Time.from_seconds(total_seconds)

    def to_seconds(self):
        return self.hour * 3600 + self.minute * 60 + self.second

    @classmethod
    def from_seconds(cls, seconds):
        hour, seconds = divmod(seconds, 3600)
        minute, seconds = divmod(seconds, 60)
        return cls(hour, minute, seconds)

    def __str__(self):
        return f"{self.hour:02}:{self.minute:02}:{self.second:02}"
```

In this example, the `__add__()` method is defined to allow two `Time` objects to be added together using the `+` operator. The `to_seconds()` method converts a `Time` object to seconds, and the `from_seconds()` method creates a new `Time` object from a given number of seconds.

Here's an example of how to create and add two `Time` objects:

```
time1 = Time(3, 45, 15)
time2 = Time(1, 20, 30)

total_time = time1 + time2
print(total_time)  # Output: "05:05:45"
```

In this example, we create two `Time` objects (`time1` and `time2`) with different hours, minutes, and seconds. We then add them together using the `+` operator, which calls the `__add__()` method we defined earlier. The resulting `total_time` object represents the sum of the two `Time` objects, which we can print out using the `__str__()` method we defined earlier.

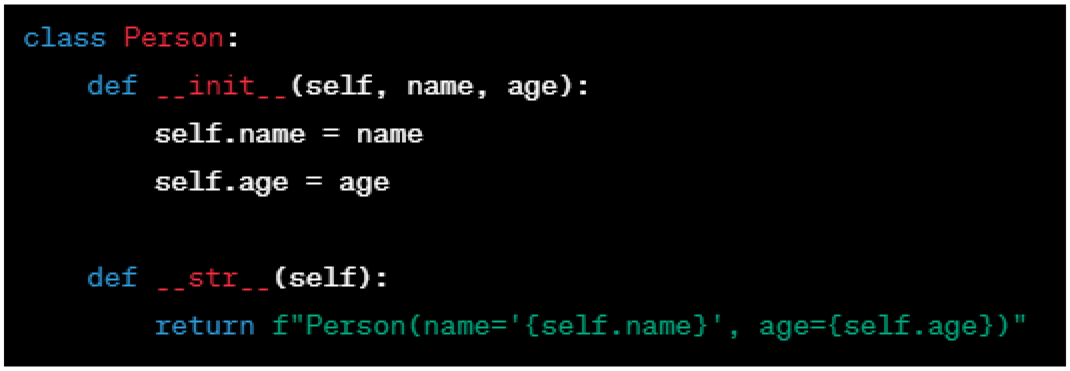
20. Discuss __str__() and __init__() methods used in class definition.

The `__init__()` method is called when an object of the class is created. It is used to initialize the attributes of the object with the values passed to it as arguments. For example:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

(In this example, the `__init__()` method is defined with `self`, `name`, and `age` as arguments. When an object of the `Person` class is created, the `__init__()` method is automatically called and the `name` and `age` attributes are initialized with the values passed to it.)

The `__str__()` method is used to return a string representation of the object. It is called when `str()` function is applied to the object or when `print()` function is used to print the object. For example:



```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"Person(name='{self.name}', age={self.age})"
```

(In this example, the `__str__()` method is defined to return a string representation of the `Person` object. When the `str()` function or `print()` function is used to print the object, the `__str__()` method is called to return the string representation.)

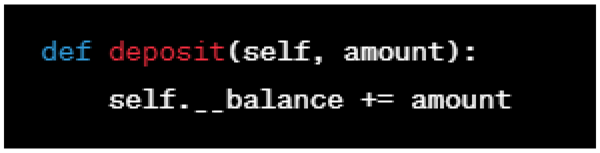
NOTE:~ The “( )” content is for understanding purpose and is not required to be written in the answer.

21. What is Encapsulation? Discuss with an example in which access specifiers are used in class definition.

Encapsulation is one of the fundamental concepts of object-oriented programming (OOP) that involves bundling data and methods that operate on that data within a single unit, called a class. The primary purpose of encapsulation is to hide the implementation details of an object from the outside world and to provide a public interface for interacting with the object.

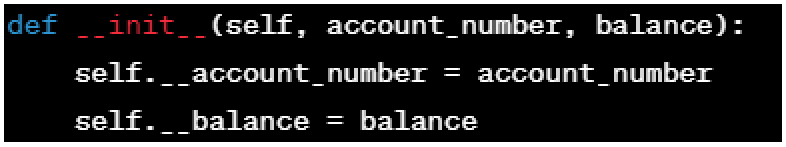
In Python, encapsulation can be achieved using access specifiers, which are special keywords that control the visibility of attributes and methods of a class. There are two types of access specifiers in Python:

1. **Public Access Specifiers:** Attributes and methods that are not prefixed with any underscore are considered public and can be accessed from anywhere in the program.



```python
def deposit(self, amount):
    self.__balance += amount
```

2. **Private Access Specifiers:** Attributes and methods that are prefixed with double underscore (__) are considered private and can only be accessed from within the class.



```python
def __init__(self, account_number, balance):
    self.__account_number = account_number
    self.__balance = balance
```

22. What is a class diagram? Create empty class and corresponding class diagram for following statements (i) class A derives from class B and Class C (ii) Class D derived from Class A defined in statement (i)

Answer not found**

23. Discuss polymorphism and demonstrate with and python program.

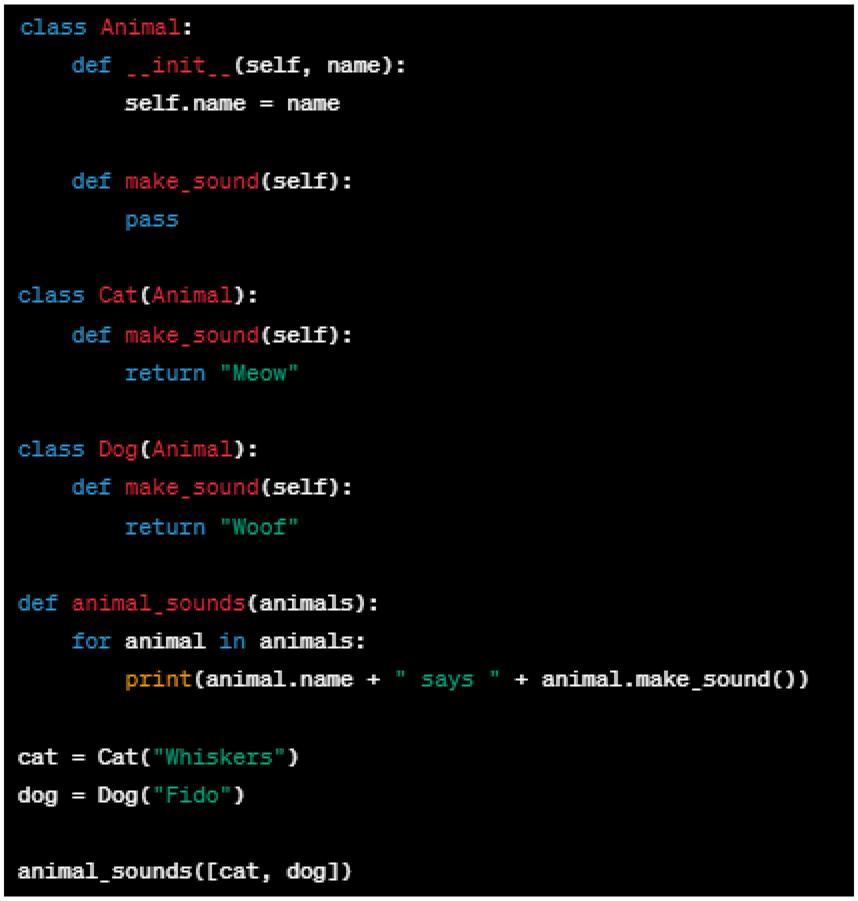
Polymorphism is the ability of objects of different classes to be used interchangeably, as they can respond to the same message or method call in different ways. In other words, polymorphism allows different classes to have the same interface or method signature, but with different implementations.

In Python, polymorphism is achieved through method overriding and method overloading.

Method overriding is when a subclass provides a different implementation of a method that is already defined in its superclass. This allows the subclass to inherit the method from its superclass but also tailor it to its specific needs.

Method overloading is when a class defines multiple methods with the same name but different parameters. This allows the class to handle different argument types and numbers in a consistent manner.

Here's an example Python program that demonstrates polymorphism:

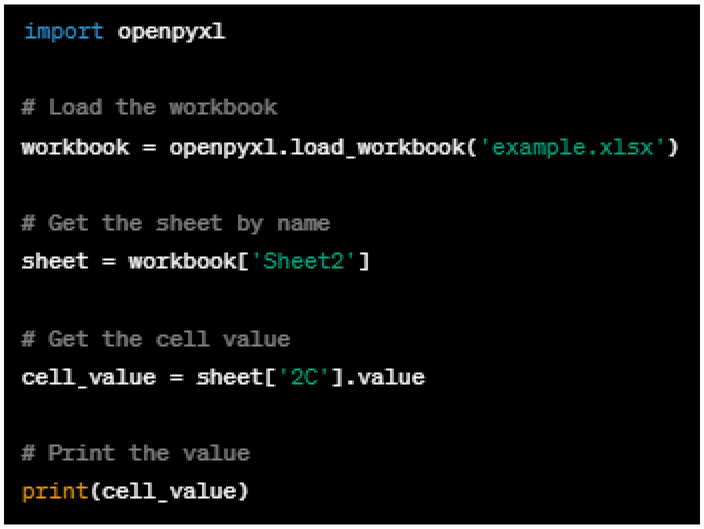


```python
class Animal:
    def __init__(self, name):
        self.name = name

    def make_sound(self):
        pass

class Cat(Animal):
    def make_sound(self):
        return "Meow"

class Dog(Animal):
    def make_sound(self):
        return "Woof"

def animal_sounds(animals):
    for animal in animals:
        print(animal.name + " says " + animal.make_sound())

cat = Cat("Whiskers")
dog = Dog("Fido")

animal_sounds([cat, dog])
```

(In this example, we have a base class `Animal` with an abstract method `make_sound`. We also have two subclasses, `Cat` and `Dog`, which inherit from `Animal` and override the `make_sound` method to provide their own implementation.The `animal_sounds` function takes a list of `Animal` objects as input and calls their `make_sound` method to produce their respective sounds. Since both `Cat` and `Dog` classes have implemented the `make_sound` method, they can be used interchangeably and produce different sounds when the function is called.This is an example of polymorphism in action, where different objects of different classes can be used interchangeably in the same context.)

24. **Write python program to read cell 2C from sheet 2 of workbook**

To read cell 2C from sheet 2 of a workbook using Python, we can use the openpyxl module. Here's an example program:

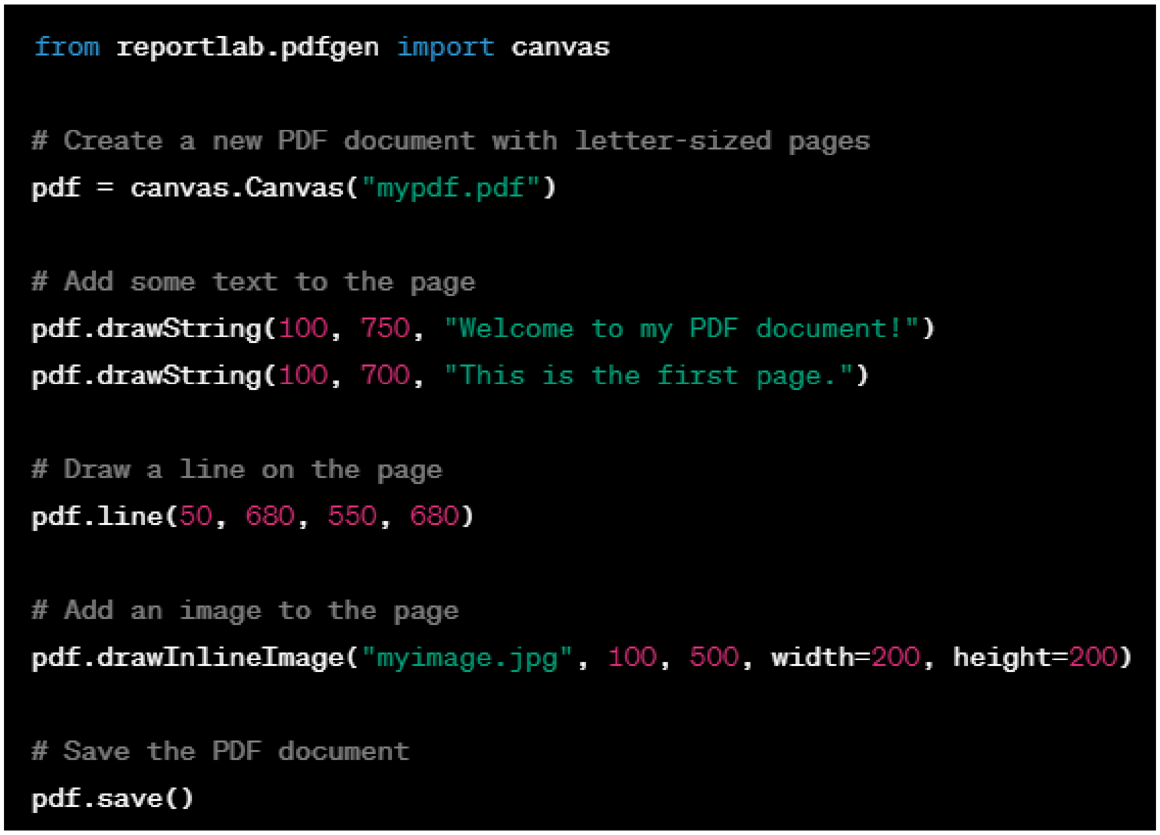


```
import openpyxl

# Load the workbook
workbook = openpyxl.load_workbook('example.xlsx')

# Get the sheet by name
sheet = workbook['Sheet2']

# Get the cell value
cell_value = sheet['2C'].value

# Print the value
print(cell_value)
```

25. **Explain how pdf pages would created in a pdf document with example.**

DF pages can be created using a variety of software tools and programming libraries. Here, I'll provide an example of how to create PDF pages using Python's `reportlab` library.

`reportlab` is a Python library that allows you to create PDF documents from scratch or modify existing ones. To create a PDF page with this library, you need to create a canvas object and then add various elements to it, such as text, shapes, and images.

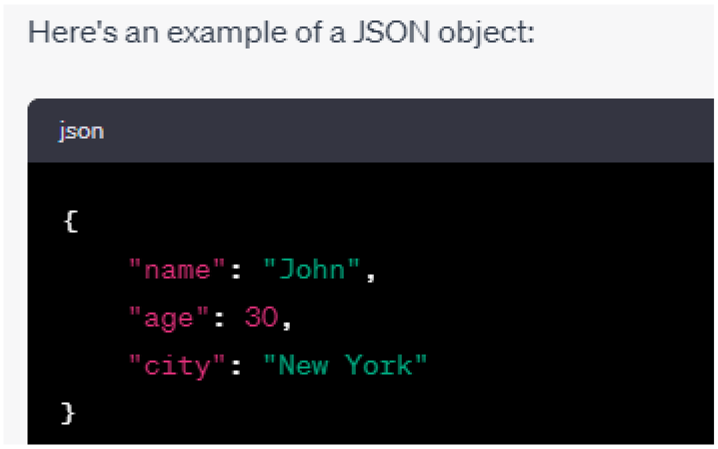


```python
from reportlab.pdfgen import canvas

# Create a new PDF document with letter-sized pages
pdf = canvas.Canvas("mypdf.pdf")

# Add some text to the page
pdf.drawString(100, 750, "Welcome to my PDF document!")
pdf.drawString(100, 700, "This is the first page.")

# Draw a line on the page
pdf.line(50, 680, 550, 680)

# Add an image to the page
pdf.drawInlineImage("myimage.jpg", 100, 500, width=200, height=200)

# Save the PDF document
pdf.save()
```

26. **What is JSON? discuss with example. Compare it with dictionary.**

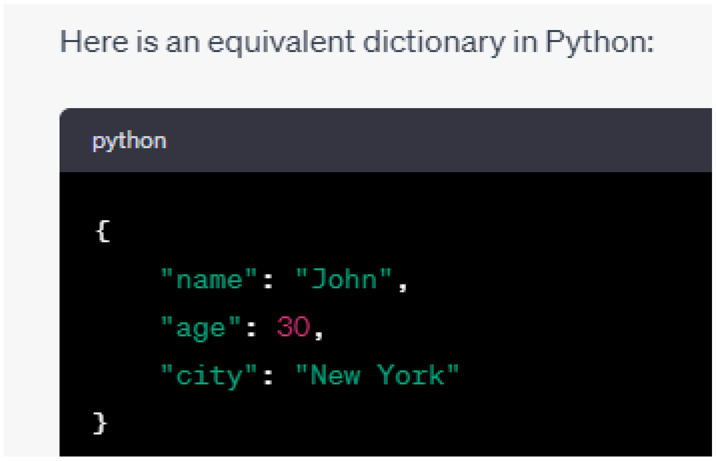
JSON stands for JavaScript Object Notation, which is a lightweight data interchange format. It is easy for humans to read and write, and for machines to parse and generate. JSON is a text format that is completely language-independent but uses conventions that are familiar to programmers.

A JSON data structure consists of key-value pairs, which are similar to Python dictionaries. However, JSON has a stricter syntax and is typically used for data exchange between different programming languages or systems.


Here's an example of a JSON object:

json

```json
{
    "name": "John",
    "age": 30,
    "city": "New York"
}
```


Here is an equivalent dictionary in Python:

python

```python
{
    "name": "John",
    "age": 30,
    "city": "New York"
}
```

27. **Compare and contrast Excel and CSV files.**

Excel and CSV (Comma-Separated Values) files are two types of file formats that are commonly used for storing and manipulating data. While there are some similarities between the two formats, there are also some important differences that distinguish them from each other.

Excel files are spreadsheet files created by Microsoft Excel or other spreadsheet software. They can contain multiple sheets, and each sheet can have multiple cells, columns, and rows. Excel files can be saved in various formats, including

XLS, XLSX, and CSV. Excel files can contain various data types, including text, numbers, dates, and formulas. Excel files also support formatting and styling of cells, including font size, color, and alignment.

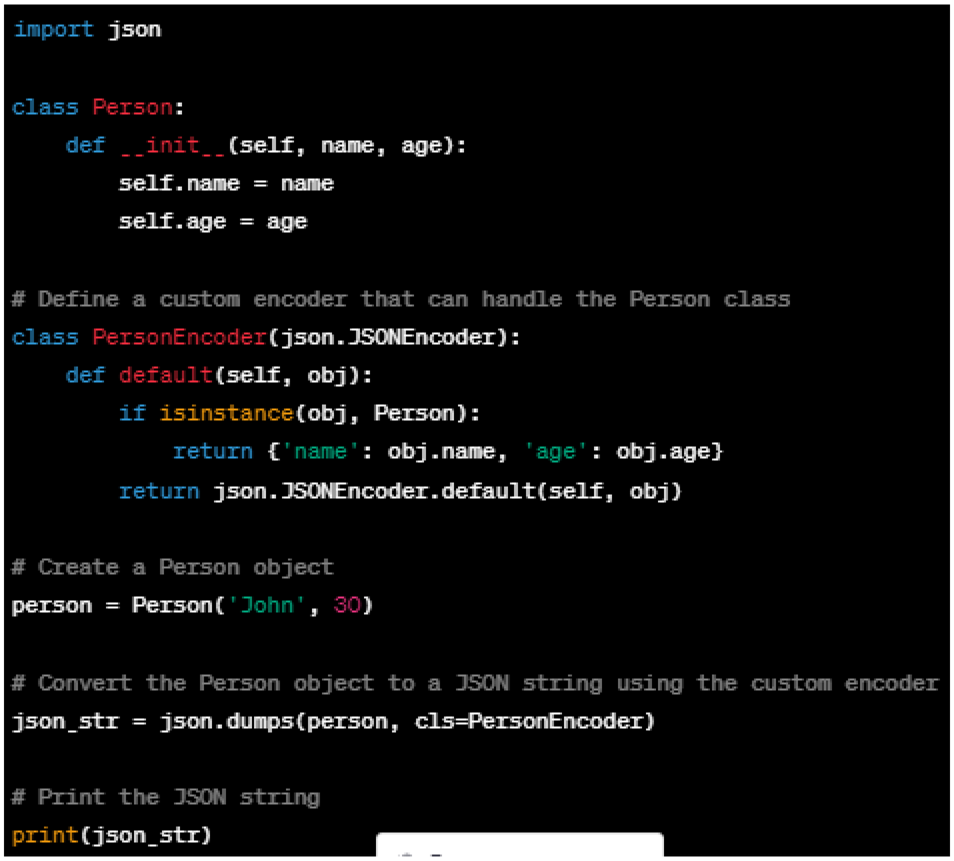
CSV files, on the other hand, are simple text files that contain data in a tabular format. Each row of data is represented by a line in the file, and each cell is separated by a comma or another delimiter. CSV files do not support formatting or styling of cells, and they can only contain simple data types such as text and numbers.

One of the main advantages of CSV files over Excel files is their simplicity and compatibility with a wide range of software and platforms. CSV files can be easily opened and edited in a text editor or spreadsheet software, and they can be imported into databases or other programs that support tabular data.

Excel files, on the other hand, offer more advanced features such as formulas, macros, and formatting, which make them more suitable for complex data analysis and reporting tasks. Excel files can also contain charts, graphs, and other visualizations that help to communicate the data more effectively.

28. Demonstrate how a Class would be converted into JSON object with an example.

To convert a class object to a JSON object, we can use the `json` module in Python. We need to define a custom encoder that converts the object to a JSON-serializable format. Here's an example:



```python
import json

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

# Define a custom encoder that can handle the Person class
class PersonEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, Person):
            return {'name': obj.name, 'age': obj.age}
        return json.JSONEncoder.default(self, obj)

# Create a Person object
person = Person('John', 30)

# Convert the Person object to a JSON string using the custom encoder
json_str = json.dumps(person, cls=PersonEncoder)

# Print the JSON string
print(json_str)
```

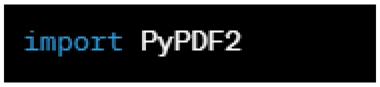
(In this example, we define a `Person` class with `name` and `age` attributes. We also define a custom encoder `PersonEncoder` that extends the `JSONEncoder` class and overrides the `default` method to handle the `Person` class. The `default` method checks if the object is an instance of `Person`, and if so, returns a dictionary with the `name` and `age` attributes. Finally, we create a `Person` object, and use `json.dumps()` to convert it to a JSON string using our custom encoder.

JSON and dictionaries are similar in that they both represent data in a key-value format. However, JSON has some differences, such as requiring all keys to be strings and not allowing trailing commas. JSON also has a more standardized format, while dictionaries can have different formats depending on the implementation. Additionally, JSON is designed to be language-independent and can be used by different programming languages, while dictionaries are specific to Python.)

29. Explain how a page from different PDFs files would be merged into a new PDF file?

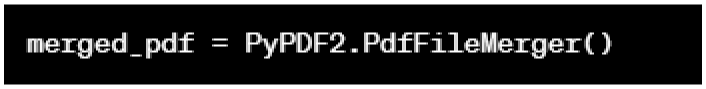
Merging pages from different PDF files into a new PDF file can be achieved using various Python libraries such as PyPDF2, PyMuPDF, etc. Here are the general steps to merge pages from different PDF files into a new PDF file using PyPDF2:

1. Import the PyPDF2 library:



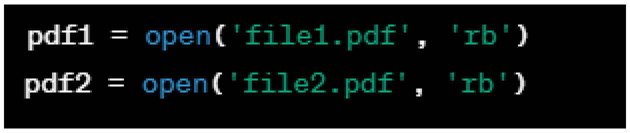
```
import PyPDF2
```

2. Import the PyPDF2 library:



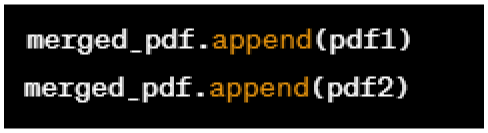
```
merged_pdf = PyPDF2.PdfFileMerger()
```

3. Open the PDF files to be merged using the `open()` method:



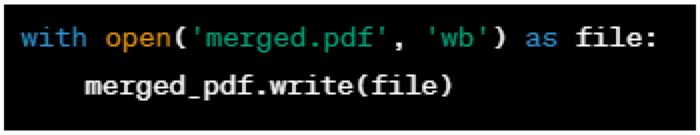
```
pdf1 = open('file1.pdf', 'rb')
pdf2 = open('file2.pdf', 'rb')
```

4. Add the pages from the PDF files to the `merged_pdf` object using the `append()` method:



```
merged_pdf.append(pdf1)
merged_pdf.append(pdf2)
```

5. Save the merged PDF file using the `write()` method:



```
with open('merged.pdf', 'wb') as file:
    merged_pdf.write(file)
```

NOTE:~ If you want you can combine all the snippets into a single program and then explain a little.

The answers may contain some errors