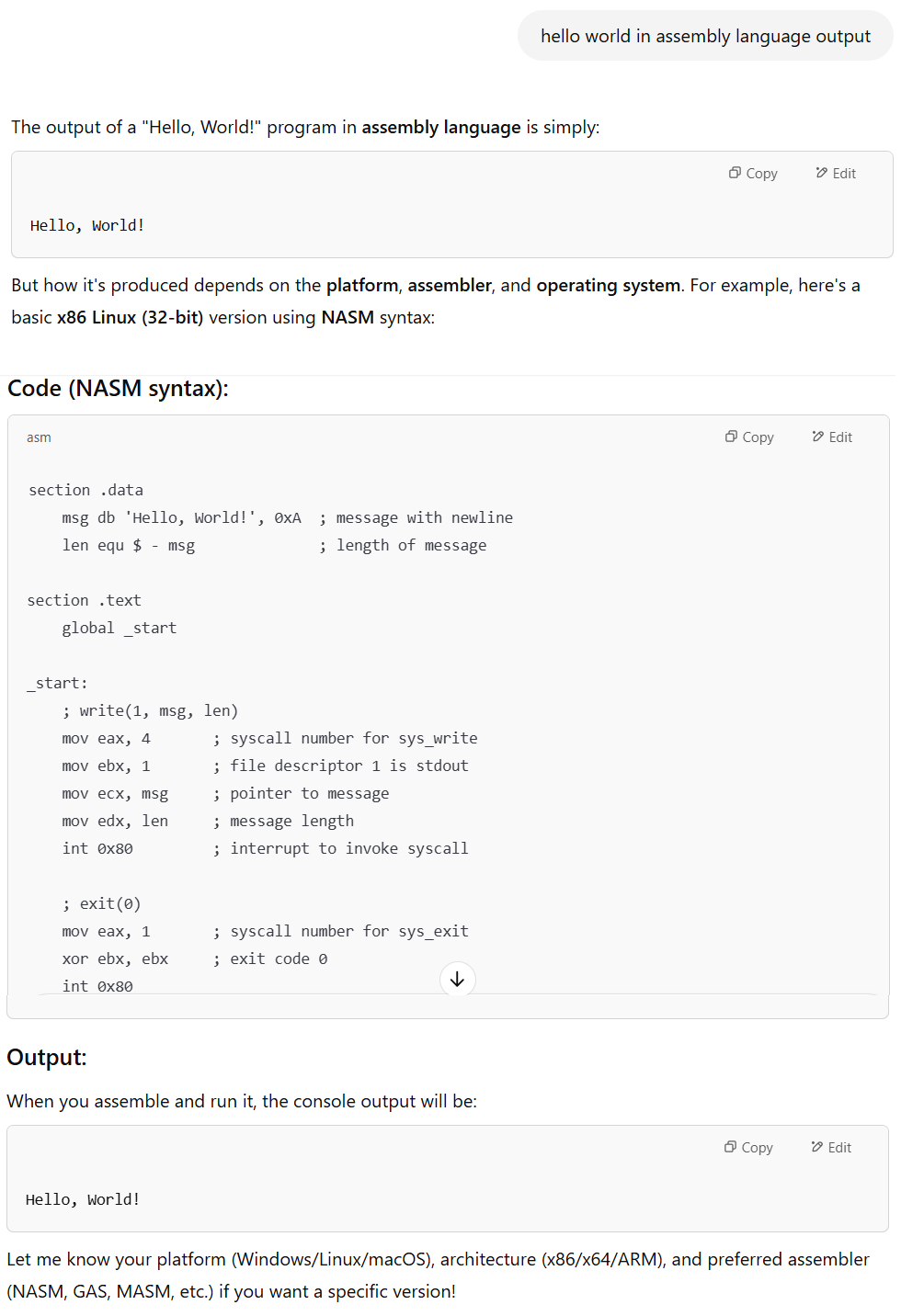
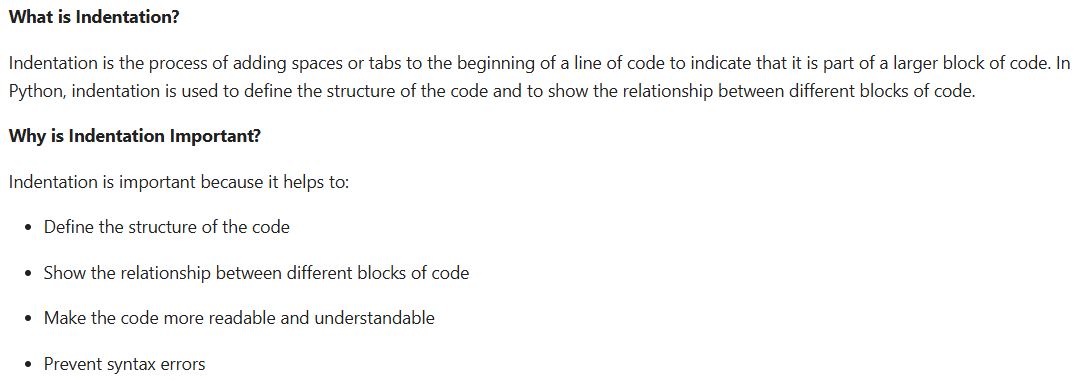
**LECTURE-02 (Sunday 19-April-2025)**

* <https://github.com/panaversity/learn-modern-ai-python/blob/main/00_python_colab/01_introduction_to_python/Agentic_AI_Python_Lesson_01_Introduction_to_Python.ipynb>
* Translator translates programming language to machine language.
* Interpreter: It is a tool that reads and executes code line by line (or statement by statement).
* Compiler: It is a tool that translates the entire source code of a program into machine code (or an intermediate code) before the program is run.
* High Level Language is designed to be easy for humans to read and write to control a computer.
* Python is a High Level Language.
* Low Level Language is closer to machine code (what the computer's hardware actually understands) and machine process low level language faster than high level language.
* ChatGPT -> write prompt -> “hello world in assembly language output”



* C language is nearest to machine language.
* “Rust” programming language is used to work on IDE.
* Rust language is also nearest to machine language.
* Default programming language is Python.
* Python is near to human language and after interpretation, it comes near to machine language.
* Quote for Java language is “write once and run anywhere”
* You write Python code (e.g., hello.py file) then Python interpreter parses and compiles code into bytecode and this bytecode is then run by the Python Virtual Machine (PVM).
* Any compatible version of Python (usually Python 3.x) should be installed in Operating System then Python code works on that Operating System.
* Python is interpreter language.
* Python is interface language.
* <https://openai.com/sora/> is best website for animation
* Application types:  
  1) Internet website -> web application  
  2) Installed in computer -> desktop application  
  3) console application  
  4) installed in mobile -> mobile application  
  5) Metaverse -> augment reality application  
  6) virtual reality -> wear glasses and entered in another world
* Python is interface language because we can develop any type of application from Python.
* There are 1.2 million jobs in Python.
* Python major rule is “Indentation” (code formatting).



Example:

My name is Abc.

  I was born in Xyz.

    I live in Pqr.

  I live near Ghi.

    My city name is Def.

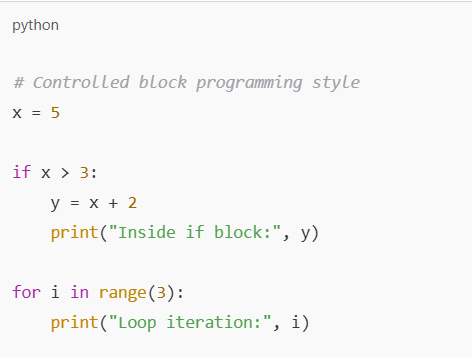
* There are two types of programming in terms of indentation

1. Linear programming is flat and sequential with minimal indentation.

Example:



1. Controlled Block (Structure) programming uses indentation to show nested blocks for decision-making and repetition.



* In Python, controlled block is identified by “:” (colon) sign.

Example 1:

for i in range(10) :

  print("Pakistan Zindabad")

Here, spaces before “print” statement is called Indentation.

Output:

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Pakistan Zindabad

Example 2:

for i in range(10):

  print("Pakistan Zindabad")

  print("Karachi")

Output:

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Example 3:

for i in range(10):

  print("Pakistan Zindabad")

  print("Karachi")

print ("end")

Output:

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

Pakistan Zindabad

Karachi

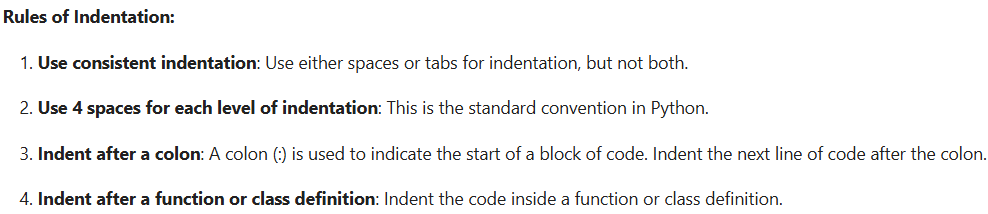
Pakistan Zindabad

Karachi

end

* Rules of Indentation

1. Rule 1: use indentation same everywhere means either use space or tab for indentation then use same indentation throughout the application. Do not do that at one place used 2 spaces and at other place, used tab.
2. Rule 2: for Python, standard indentation is 4 spaces. Also, in VS Code, 4 spaces is used for indentation by default.
3. Rule 3: As in Python, “:” (colon) sign is used to identify indentation so after colon sign, indented code should be present.
4. Rule 4: use indentation in functions and classes



Example 1:

# Correct indentation for IF block

if True:

    print("Hello, World!")

    print("This is a block of code")

Output:

Hello, World!

This is a block of code

Example 2:

# Incorrect indentation for IF block

if True:

print("Hello, World!")

  print("This is a block of code")

Output:

File ["<ipython-input-12-8cdd8191549a>"](https://localhost:8080/), line 3 print("Hello, World!") ^ IndentationError: expected an indented block after 'if' statement on line 2

Example 3:

# Correct indentation for FUNCTION block

def greet(name: str):

    print("Hello, " + name + "!")

Output: Code block successfully executed.

Example 4:

# Incorrect indentation for FUNCTION block

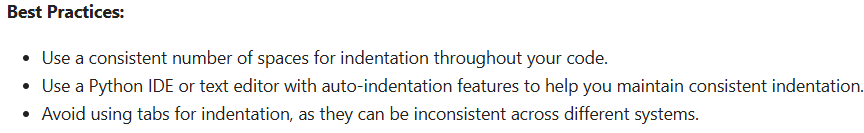
def greet(name: str):

print("Hello, " + name + "!")

Output:

File ["<ipython-input-16-1c3a6ae72619>"](https://localhost:8080/), line 3 print("Hello, " + name + "!") ^ IndentationError: expected an indented block after function definition on line 2

* Best practices for Indentation:



* For indentation in Python, “PEP 8” standards are followed.
* PEP 8 is the “Python Enhancement Proposal” that outlines the style guide for Python code. It's essentially the rulebook for writing clean, readable, and consistent Python code.
* Key principles of PEP 8:

1. Indentation: Use 4 spaces per indentation level (not tabs).
2. Line Length: Limit lines to 79 characters.
3. Blank Lines: Use blank lines to separate functions, classes, and blocks inside functions for readability.
4. Imports:
   1. Should usually be on separate lines.
   2. Grouped in the order: standard library, third-party libraries, and local application imports.
5. Whitespace:
   1. Avoid extra spaces in expressions and statements, e.g. x = 1 not x=1.
   2. No space before a comma, and one space after it.
6. Naming Conventions:
   1. Variables, functions: lower\_case\_with\_underscores (snake\_case)
   2. Classes: CapWords (PascalCase)
   3. Constants: ALL\_CAPS
7. Function and Variable Annotations: Use type hints if applicable.
8. Docstrings: Use triple quotes for documenting functions, classes, and modules.

* Naming conventions:

1. camelCase
2. snake\_case
3. PascalCase

* Datatypes: To store values, programming languages provides structures called Datatypes.
* There are two primary types “Text” (Quality) and “Numbers” (Quantity)
* For text, we use “string” datatype.
* For numbers, we use “integer” datatype.
* In string datatype, output of “1+1“ will be “11” while in integer datatype, output of “1+1 “ will be “2”.
* Dynamic Datatype/Python: whatever value we pass, variable datatype will be selected.

x=12

a="wow"

Here,

Integer datatype will be selected for “x” variable while string datatype will be selected for “a” variable.

* Restricted or Predefined or Static Datatype/Python: set variable datatype while initializing value.

x : int =12

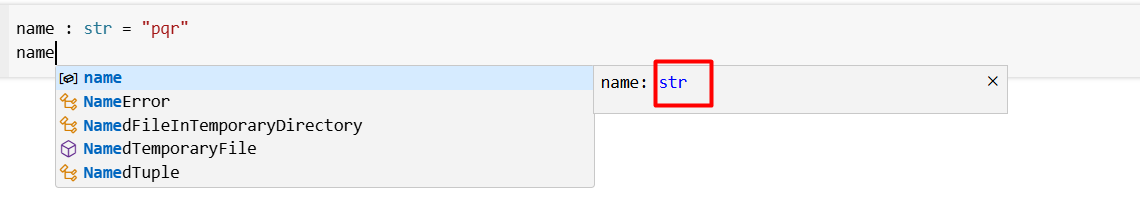
a : str ="wow"

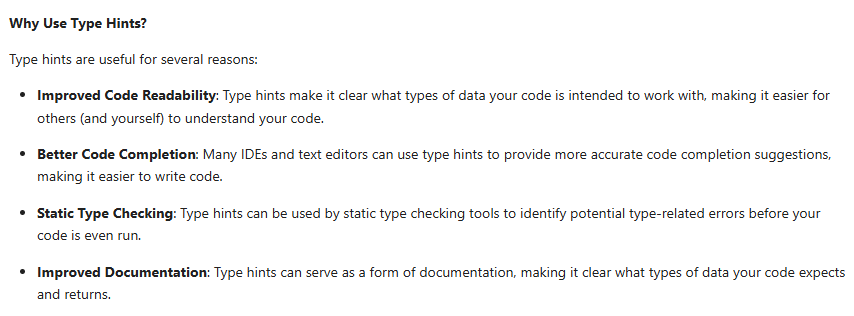
* IntelliSense is a feature found in many modern code editors and IDEs (like Visual Studio Code, PyCharm, Visual Studio, etc.) that helps you write code faster and with fewer errors by providing:

1. Auto-completion: Suggests variable names, functions, classes, modules, etc., as you type.
2. Parameter Info: Shows the expected parameters of a function or method, including data types and default values.
3. Quick Info: Displays documentation or type hints when you hover over a symbol.
4. Code Snippets: Inserts common blocks of code with a shortcut.
5. Error Checking: Highlights syntax and type errors as you write code.

* Datatype Hints: It provides hint to developers to choose datatype on the basis of value.

Example:





* When we are working in large project, we forget datatype of variable so it is best practice to use restricted or predefined data type for variables.
* Python is dynamic datatype.

x=10

print(x)

x="pq"

print(x)

x=10.37

print(x)

y : int = 20

print(y)

y = "pqr"

print(y)

z : str ="abc"

print(z)

z = 10

print(z)

Output:

10

pq

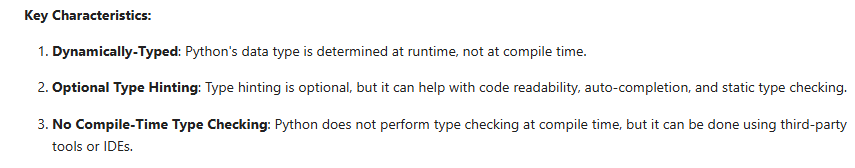
10.37

20

pqr

abc

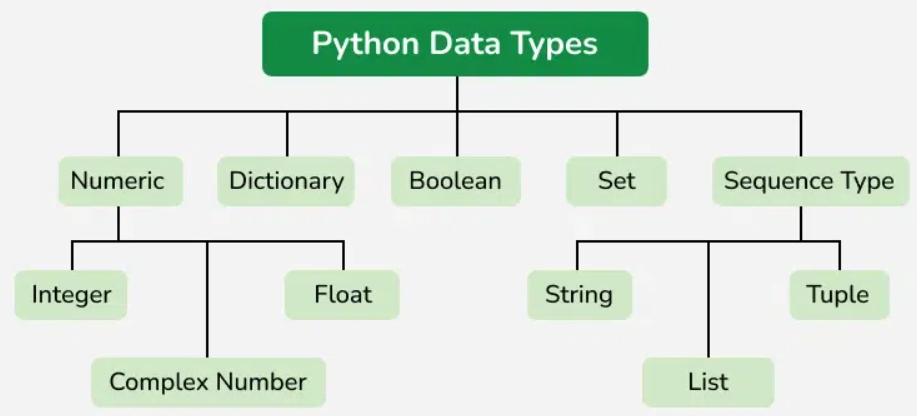
10



* <https://github.com/panaversity/learn-modern-ai-python/blob/main/00_python_colab/02_data_types/Agentic_AI_Python_Lesson_02_Data_Types.ipynb>
* Data types determine the type of value a variable can hold and the operations that can be performed on it. Datatype define the format, structure, size, range, and behavior of data, controlling how it's stored and used in a program. This helps ensure data is used correctly and efficiently.
* Natural data types are Male and Female
* Primitive datatypes are default datatypes.

1. Integers: for numbers
2. Float: for decimal numbers
3. String: for text in single or double or triple quotation
4. Date
5. Audio
6. Video
7. File

* Custom datatypes



* Integer: 10
* Float: 20.43
* Complex Number: imaginary and real numbers
* Boolean: true,flase
* Variable -> changeable
* To store data, we use variable
* Everything is object
* Everyone is object. Characteristics of Object

1. Methods: object performs action
2. Properties/Attributes: name, fathername, age of object

* Some objects have similar methods and properties
* Variable name should be proper and relevant to the value.
* Operators: +. =. -, \*, /

print(2+2)

print(5-2)

print(3\*4)

print(9/3)

Output:

4

3

12

3.0

* Operator: type of action

Operand: things on which operation will be performed

Operation: perform action

Example:

2+4

Here,

Operator is “+”.

Operands are “2” and “4”.

Operation is Addition

* “=” (Assignment Operator) is used to store value in variable.  
  name = "ABC"

age = 27

* Predefined functions are default functions:

1. “print” method is used to print data.

Example:

x : int = 10

print(x)

Output:

10

1. “type” method is used to get data type of variable.

Example:

x : int = 10

print(type(x))

Output:

<class 'int'>

1. “id” method is use to get memory address of variable.

Example:

x : int = 10

print(id(x))

Output:

10751144

1. “dir” method is used to get list of methods and attributes which that object support which is passed in argument.

Example 1:

x : int = 10

print(dir(x))

Output:

['\_\_abs\_\_', '\_\_add\_\_', '\_\_and\_\_', '\_\_bool\_\_', '\_\_ceil\_\_', '\_\_class\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_divmod\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_float\_\_', '\_\_floor\_\_', '\_\_floordiv\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getnewargs\_\_', '\_\_getstate\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_index\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_int\_\_', '\_\_invert\_\_', '\_\_le\_\_', '\_\_lshift\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_neg\_\_', '\_\_new\_\_', '\_\_or\_\_', '\_\_pos\_\_', '\_\_pow\_\_', '\_\_radd\_\_', '\_\_rand\_\_', '\_\_rdivmod\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rfloordiv\_\_', '\_\_rlshift\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_ror\_\_', '\_\_round\_\_', '\_\_rpow\_\_', '\_\_rrshift\_\_', '\_\_rshift\_\_', '\_\_rsub\_\_', '\_\_rtruediv\_\_', '\_\_rxor\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_sub\_\_', '\_\_subclasshook\_\_', '\_\_truediv\_\_', '\_\_trunc\_\_', '\_\_xor\_\_', 'as\_integer\_ratio', 'bit\_count', 'bit\_length', 'conjugate', 'denominator', 'from\_bytes', 'imag', 'numerator', 'real', 'to\_bytes']

Example 2:

print(dir(20))

Output:

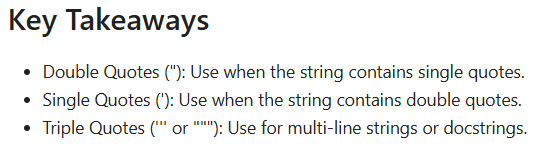
['\_\_abs\_\_', '\_\_add\_\_', '\_\_and\_\_', '\_\_bool\_\_', '\_\_ceil\_\_', '\_\_class\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_divmod\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_float\_\_', '\_\_floor\_\_', '\_\_floordiv\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getnewargs\_\_', '\_\_getstate\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_index\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_int\_\_', '\_\_invert\_\_', '\_\_le\_\_', '\_\_lshift\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_neg\_\_', '\_\_new\_\_', '\_\_or\_\_', '\_\_pos\_\_', '\_\_pow\_\_', '\_\_radd\_\_', '\_\_rand\_\_', '\_\_rdivmod\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rfloordiv\_\_', '\_\_rlshift\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_ror\_\_', '\_\_round\_\_', '\_\_rpow\_\_', '\_\_rrshift\_\_', '\_\_rshift\_\_', '\_\_rsub\_\_', '\_\_rtruediv\_\_', '\_\_rxor\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_sub\_\_', '\_\_subclasshook\_\_', '\_\_truediv\_\_', '\_\_trunc\_\_', '\_\_xor\_\_', 'as\_integer\_ratio', 'bit\_count', 'bit\_length', 'conjugate', 'denominator', 'from\_bytes', 'imag', 'numerator', 'real', 'to\_bytes']

* There are five built-in datatypes of Python.

1. Integer (int)
2. String (str)
3. Boolean (bool)

is\_here : bool = True

* String: we can use single or double quotation for values of string variable while we use triple double quotation to support multiline value of string variable.



Example:

x = "abc"

y = 'pqr'

z = """def

   ghi

         jkl

                    mno"""

print(x)

print(y)

print(z)

Output:

abc

pqr

def

ghi

jkl

mno

* Boolean datatype accepts values “True” or “False”

Example:

a : bool = True

print(a)

b = False

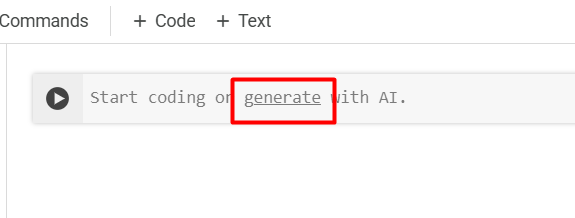
print(b)

Output:

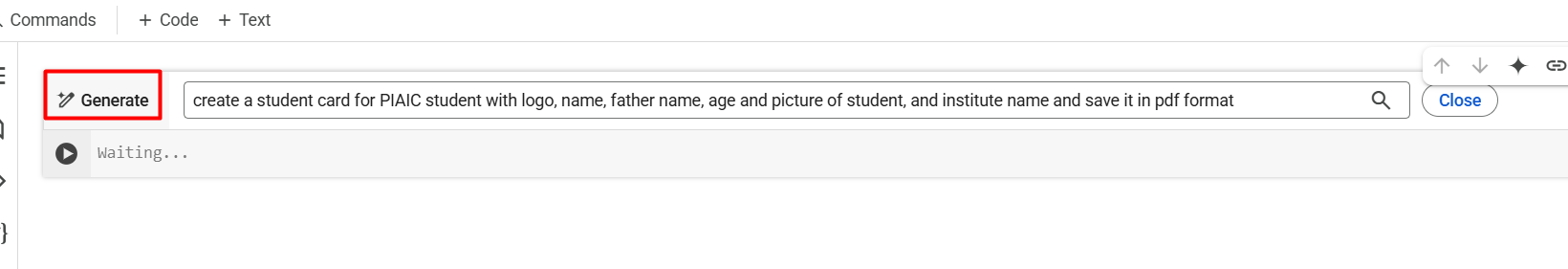
True

False

* “generate” link is used to provide prompt in text and get its code.



Pass prompt and click on “Generate” button and you will get code then you can execute code.



Example 1: Prompt : create a "4 inch x 4 inch" size student card having two line black border, for university student with left aligned university logo, right aligned picture of student and centrally aligned student information like student name, student's father name, student age and save student card in pdf"4 inch x 4 inch" file format

Code received:

# prompt: create a "4 inch x 4 inch" size student card having two line black border, for university student with left aligned university logo, right aligned picture of student and centrally aligned student information like student name, student's father name, student age and save student card in pdf"4 inch x 4 inch" file format

from reportlab.lib.pagesizes import inch, landscape

from reportlab.pdfgen import canvas

from reportlab.lib.utils import ImageReader

from PIL import Image

def create\_student\_card(university\_logo\_path, student\_image\_path, student\_name, father\_name, student\_age, output\_filename):

    width, height = 4 \* inch, 4 \* inch

    c = canvas.Canvas(output\_filename, pagesize=(width, height))

    # Draw borders

    c.setStrokeColorRGB(0, 0, 0)  # Black color

    c.setLineWidth(2)  # 2-point line width

    c.rect(0, 0, width, height)

    # Add university logo (left-aligned)

    try:

      university\_logo = ImageReader(Image.open(university\_logo\_path))

      c.drawImage(university\_logo, 0.2 \* inch, 2.5 \* inch, width=0.7 \* inch, height=0.7 \* inch, preserveAspectRatio=True)

    except FileNotFoundError:

      print("University logo file not found.")

    # Add student picture (right-aligned)

    try:

        student\_picture = ImageReader(Image.open(student\_image\_path))

        c.drawImage(student\_picture, width - 1 \* inch, 2.5 \* inch, width=0.7 \* inch, height=0.7 \* inch, preserveAspectRatio=True)

    except FileNotFoundError:

        print("Student image file not found.")

    except Exception as e:

      print(f"Error processing image: {e}")

    # Add student information (centered)

    c.setFont("Helvetica", 10)

    text\_y = 2.0 \* inch

    c.drawCentredString(width / 2, text\_y, f"Name: {student\_name}")

    text\_y -= 0.2 \* inch

    c.drawCentredString(width / 2, text\_y, f"Father's Name: {father\_name}")

    text\_y -= 0.2 \* inch

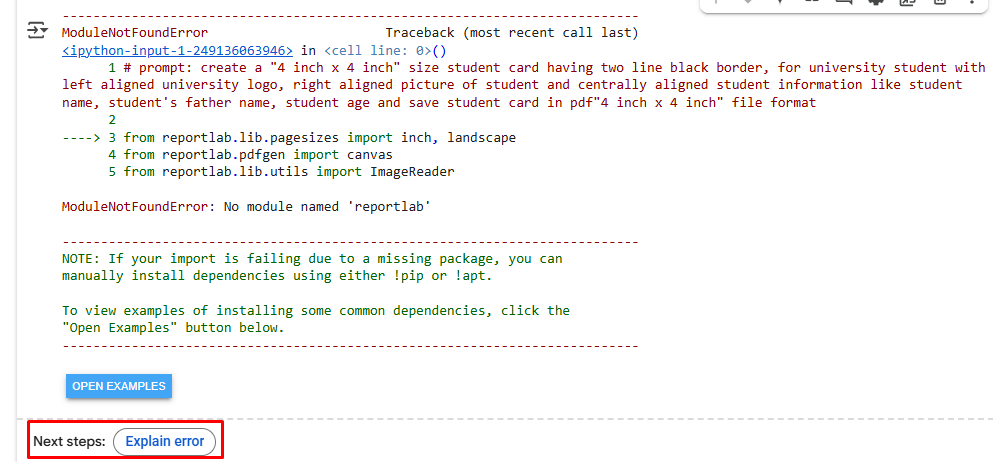
    c.drawCentredString(width / 2, text\_y, f"Age: {student\_age}")

    c.save()

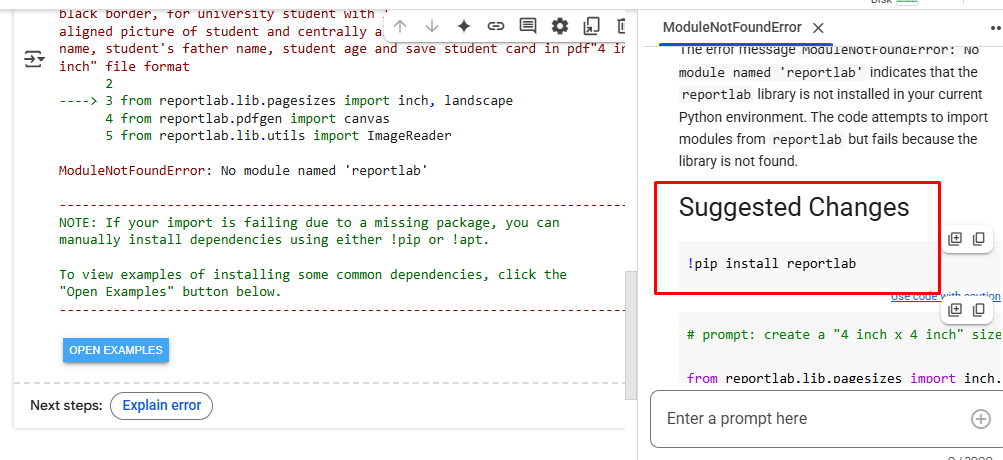
# Example usage:

create\_student\_card("university\_logo.png", "student\_photo.jpg", "John Doe", "David Doe", 20, "student\_card.pdf")

When code executed, error raised.



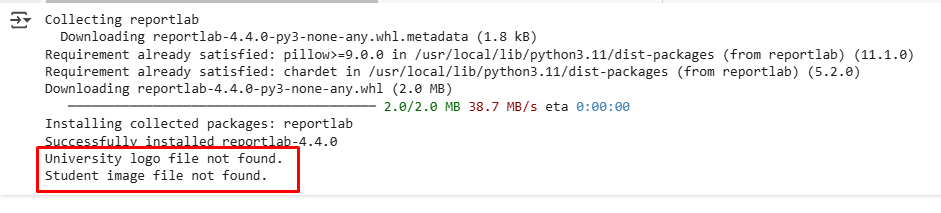
When clicked on “Explain error” button, a new tab opened at left side which shows information related to error together with suggested change.



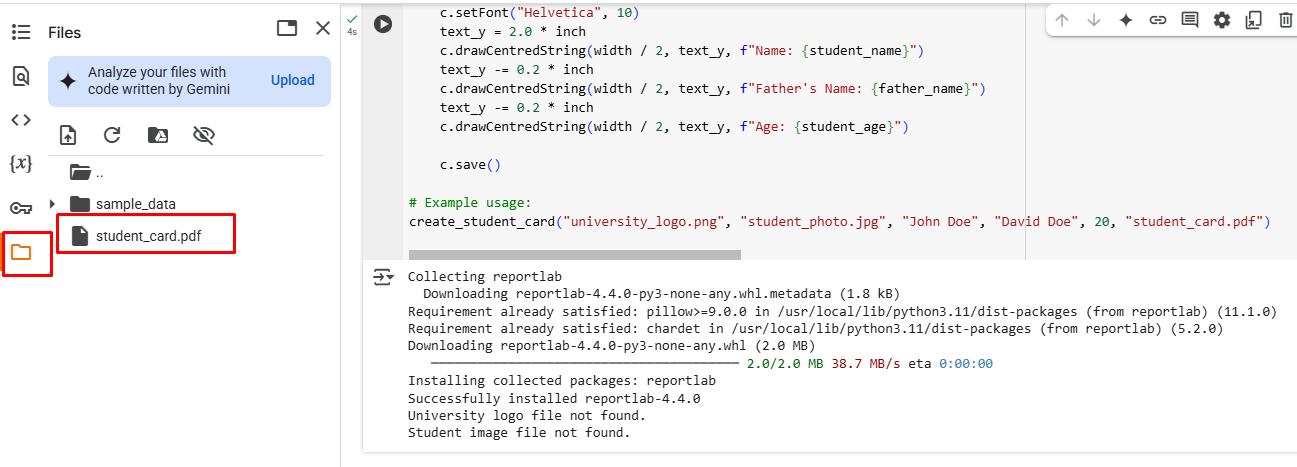
So, added suggested below change at top of received code:

!pip install reportlab

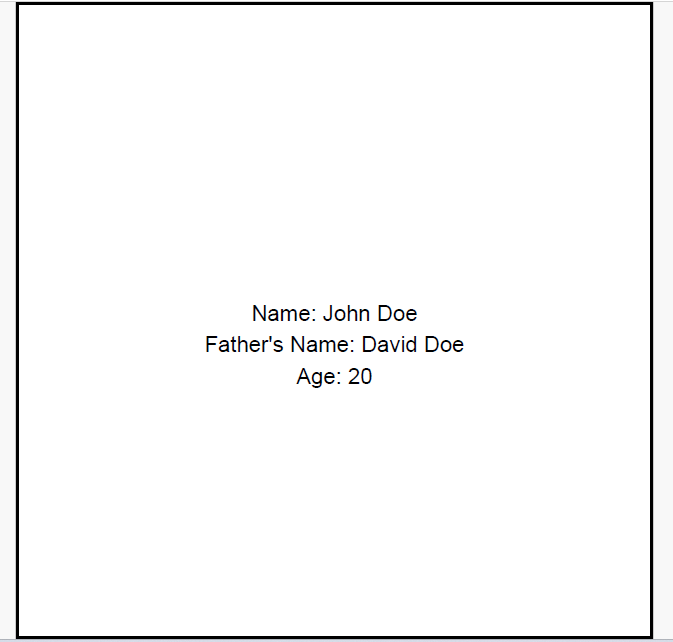
Then executed code, code executed successfully and get warning message which is highlighted in below image:



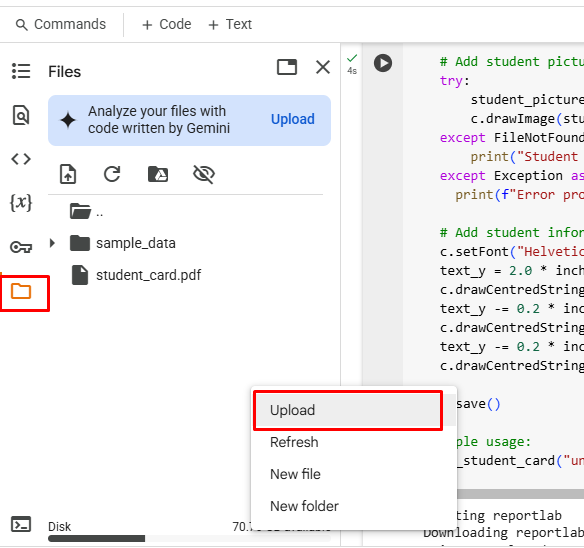
And found that “student\_card.pdf” is generated.



When double clicked at “student\_card.pdf” file, file download in “Downloads” folder of laptop. Ouput of generated “student\_card.pdf” file:

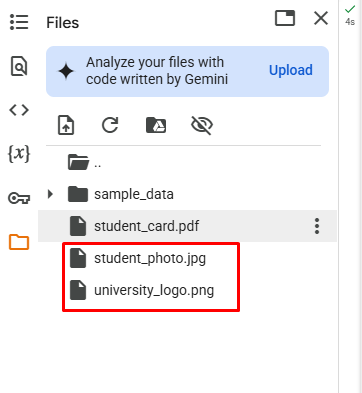


Files uploaded by:

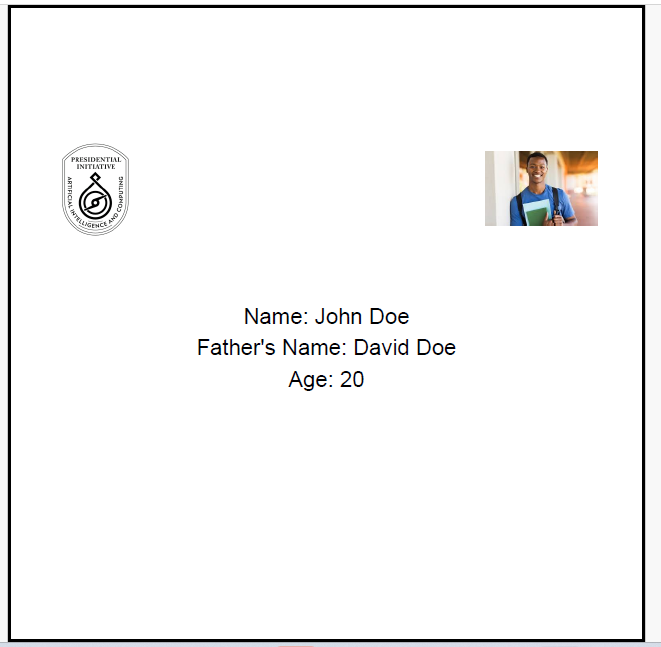


Then uploaded logo of university and picture of student files after renaming to “university\_logo.png” and “student\_photo.jpg” due to received code contains these filename:





Then executed code again and “student\_card.pdf” file updated as:



After that made some changes to adjust student card, updated code is as:

# prompt: create a "4 inch x 4 inch" size student card having two line black border, for university student with left aligned university logo, right aligned picture of student and centrally aligned student information like student name, student's father name, student age and save student card in pdf"4 inch x 4 inch" file format

!pip install reportlab

from reportlab.lib.pagesizes import inch, landscape

from reportlab.pdfgen import canvas

from reportlab.lib.utils import ImageReader

from PIL import Image

def create\_student\_card(university\_logo\_path, student\_image\_path, student\_name, father\_name, student\_age, output\_filename):

    width, height = 4 \* inch, 1 \* inch

    c = canvas.Canvas(output\_filename, pagesize=(width, height))

    # Draw borders

    c.setStrokeColorRGB(0, 0, 0)  # Black color

    c.setLineWidth(2)  # 2-point line width

    c.rect(0, 0, width, height)

    # Add university logo (left-aligned)

    try:

      university\_logo = ImageReader(Image.open(university\_logo\_path))

      c.drawImage(university\_logo, 0.2 \* inch, 0.1 \* inch, width=0.7 \* inch, height=0.7 \* inch, preserveAspectRatio=True)

    except FileNotFoundError:

      print("University logo file not found.")

    # Add student picture (right-aligned)

    try:

        student\_picture = ImageReader(Image.open(student\_image\_path))

        c.drawImage(student\_picture, width - 1 \* inch, 0.1 \* inch, width=0.7 \* inch, height=0.7 \* inch, preserveAspectRatio=True)

    except FileNotFoundError:

        print("Student image file not found.")

    except Exception as e:

      print(f"Error processing image: {e}")

    # Add student information (centered)

    c.setFont("Helvetica", 10)

    text\_y = 0.6 \* inch

    c.drawCentredString(width / 2, text\_y, f"Name: {student\_name}")

    text\_y -= 0.2               \* inch

    c.drawCentredString(width / 2, text\_y, f"Father's Name: {father\_name}")

    text\_y -= 0.2 \* inch

    c.drawCentredString(width / 2, text\_y, f"Age: {student\_age}")

    c.save()

# Example usage:

create\_student\_card("university\_logo.png", "student\_photo.jpg", "John Doe", "David Doe", 20, "student\_card.pdf")

Output “student\_card.pdf” file is as:

