2025 Batch

CSE/IT Polytechnic

Python

Polytechnic

ONE SHOT

Syllabus:

- Brief History of Python,
- Python Versions
- Installing Python
- Environment Variables
- Executing Python from the Command Line IDLE,
- Editing Python Files Python Documentation
- Getting Help
- Dynamic Types Python. Reserved Words
- Naming Conventions

Python language का इतिहास (History of Python language):

The Beginnings – Late 1980s

- **Creator:** Python was created by Guido van Rossum, a Dutch programmer, in the late 1980s.
- Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands.
- **Inspiration:** Guido wanted to create a language that was simple and accessible, inspired by his work on the ABC programming language.
- ABC programming language was primarily developed to run on the Amoeba operating system, a distributed operating system, and Unix-based systems.

Python language का इतिहास (History of Python language):

शुरुआती दौर – 1980 का दशक

- **निर्माता(Creator):** Python का निर्माण डच प्रोग्रामर गुइडो वैन रोसम द्वारा 1980 के दशक के अंत में किया गया था।
- गुइडो वैन रोसम ने इसका एप्लिकेशन आधारित कार्य दिसंबर 1989 में सेंटरम विस्कुंडे & इन्फॉर्मेटिका (CWI) में शुरू किया, जो नीदरलैंड्स में स्थित है।
- **प्रेरण(Inspiration):** गुइडो एक ऐसी भाषा बनाना चाहते थे जो सरल और सुलभ(accessible) हो, और इसकी प्रेरणा उन्हें ABC प्रोग्रामिंग भाषा पर अपने काम से मिली।
- ABC प्रोग्रामिंग भाषा को मुख्य रूप से Amoeba ऑपरेटिंग सिस्टम (जो एक वितरित ऑपरेटिंग सिस्टम है) और Unix-आधारित सिस्टम्स पर चलाने के लिए विकसित किया गया था।

The Name "Python"

- Guido named the language "Python" not after the snake, but after the British comedy show "Monty Python's Flying Circus" because he wanted a fun and unique name.

- Python's first public release, Python 0.9.0, was launched on February 20, 1991. This version introduced several core features of Python, including functions, exception handling, and key data types like lists, strings, and dictionaries.

- Later, the first official version, Python 1.0, was released in January 1994.

नाम "Python"

- गुइडो ने इस भाषा का नाम "Python" साँप के नाम पर नहीं रखा, बल्कि ब्रिटिश कॉमेडी शो "Monty Python's Flying Circus" के नाम पर रखा क्योंकि वह एक मजेदार और अनोखा नाम चाहते थे।

- Python का पहला public release, Python 0.9.0, 20 फरवरी 1991 को जारी किया गया। इस संस्करण में Python की कई मुख्य विशेषताएँ शामिल थीं, जैसे फ़ंक्शंस, Error संभालने की सुविधा (Exception Handling), और लिस्ट, स्ट्रिंग्स, डिक्शनरी जैसे मुख्य डेटा टाइप्स।

- इसके बाद, Python का first official version, Python 1.0, जनवरी 1994 में जारी किया गया।

Python version :

1. Python 0.9.0 (February 1991)

- The first release of Python, developed by Guido van Rossum.
- Introduced basic features like functions, exception handling, and modules.

2. Python 1.x Series (January 1994 – October 2000)

- Python 1.0 was officially released in January 1994.
- 1.5.2 (1999) was the last significant version in the 1.x series.
- Introduced core features like list comprehensions.
- Fixed many bugs and performance issues..

1.5.3 →
1.5.4 →

1.x, 2.x, 3.x

first

current

3. Python 2.x Series (October 2000 – January 2020)

- Python 2 was released with backward compatibility with Python 1, but it introduced major enhancements.

1.x 2.x

Python 2.0 (October 2000):

- List comprehensions.
- Garbage collection.
- Unicode support.

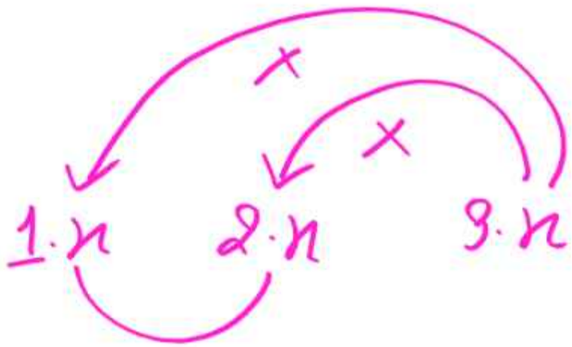
Python 2.1 (April 2001) to Python 2.7 (July 2010):

- Gradual improvements in performance, error handling, and library extensions.
- Python 2.7 was the final version of Python 2, introduced many features from Python 3.x (like dictionary comprehensions and new-style classes).

Python 2.7 reached its end of life on January 1, 2020.

4. Python 3.x Series (December 2008 – Present)

- Python 3 was introduced to fix the limitations of Python 2, though it was not backward-compatible.
- Python 3.0 (December 2008)
- Python 3.1 (June 2009)
- Python 3.2 (February 2011)
- Python 3.3 (September 2012)
- Python 3.4 (March 2014)
- Python 3.5 (September 2015)
- Python 3.6 (December 2016)
- Python 3.7 (June 2018)
- Python 3.8 (October 2019)
- Python 3.9 (October 2020)
- Python 3.10 (October 2021)
- Python 3.11 (October 2022)
- Python 3.12 (October 2023)
- Python is 3.14.0, released on October 7, 2025



2.x → Print "Hello"; → long data-type

3.x → Print ("Hello"); 3x not support.

Introduction to Python Progamming:

- Python is a high-level, interpreted, general-purpose and dynamic programming programming language known for its readability, simplicity, and flexibility.

- It was created by Guido van Rossum and first released in 1991. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

- Python is widely used in web development, data analysis, artificial intelligence, scientific computing, automation, and many other fields.

Assembly → low-level lang – (Platform dependent)

Python, java → High-level Pro. lang →

C-Pro →

int a=10;

a=10;

b=20.28;

Python प्रोग्रामिंग का परिचय:

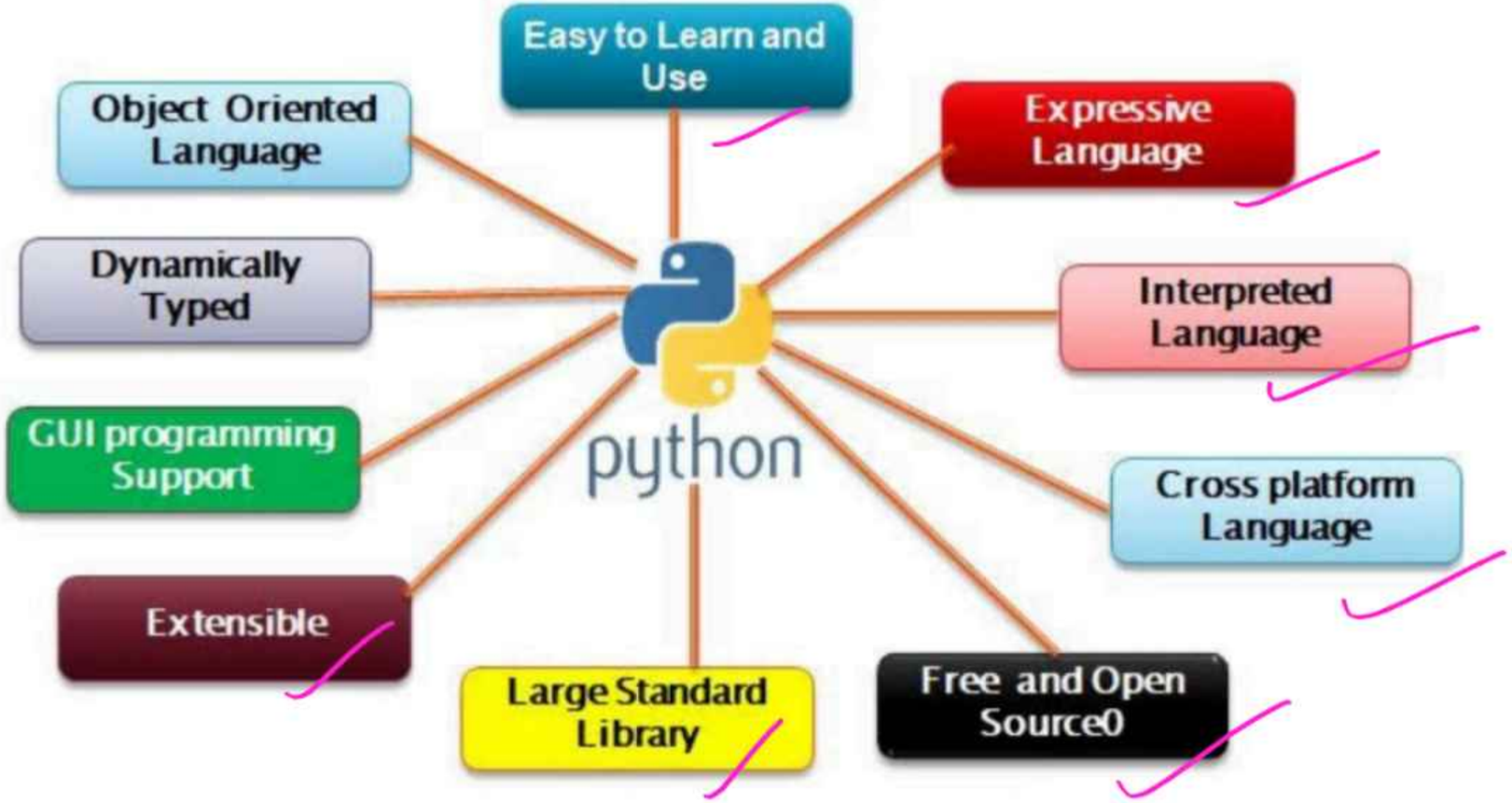
- Python एक उच्च-स्तरीय (high-level), इंटरप्रेटेड (interpreted), और जनरल-पर्पस (general-purpose) प्रोग्रामिंग भाषा है, जो अपनी सरलता, साफ-सुथरे कोड, और लचीलापन के लिए जानी जाती है।

- Python को गाइडो वैन रोसुम ने बनाया था, और इसे पहली बार 1991 में रिलीज किया गया। यह भाषा कई तरह की प्रोग्रामिंग शैलियों(paradigms) का समर्थन करती है, जैसे प्रक्रियात्मक (procedural), ऑब्जेक्ट-ओरिएंटेड (object-oriented), और फंक्शनल प्रोग्रामिंग (functional programming)।

- Python का उपयोग वेब डेवलपमेंट, डेटा विश्लेषण(analysis), आर्टिफिशियल इंटेलिजेंस, वैज्ञानिक गणना(calculations), ऑटोमेशन, और कई अन्य क्षेत्रों में बड़े पैमाने पर किया जाता है।

Python Programming Features:

- The main features of Python programming are:
- Python प्रोग्रामिंग की मुख्य विशेषताएँ इस प्रकार हैं:



1.Simple and readable syntax:

- Python's syntax is easy to read and write, making it accessible to beginners and professionals a like.
- Python का कोड लिखना और समझना आसान है, जिससे इसे शुरुआती(beginners) और पेशेवर(professionals) दोनों लोग आसानी से सीख सकते हैं।

2. Interpreted language:

- Python code is executed line by line, without the need for prior compilation, which speeds up debugging and development.
- Python का कोड सीधे चलाया जा सकता है, बिना पहले कंपाइल करने के, जिससे डिबगिंग और विकास तेज़ हो जाता है।

3. General-purpose language:

- Python can be used in a variety of fields, such as web development, data science, machine learning, automation, etc.
- Python का उपयोग कई क्षेत्रों में किया जा सकता है, जैसे वेब डेवलपमेंट, डेटा साइंस, मशीन लर्निंग, और ऑटोमेशन।

4. Cross-platform compatibility:

- Python code runs on different operating systems (Windows, macOS, Linux) without modification.
- Python का कोड विभिन्न ऑपरेटिंग सिस्टम (Windows, macOS, Linux) पर बिना बदलाव के चल सकता है।

5. Large library and framework support:

- Python has a vast collection of libraries and frameworks like NumPy, Pandas, Django, and Flask, making programming easier faster.
- Python के पास बड़ी संख्या में लाइब्रेरी और फ्रेमवर्क (जैसे NumPy, Pandas, Django, Flask) हैं, जो काम को आसान और तेज़ बनाते हैं।

6. Supports multiple programming paradigms:

- Python supports procedural, object-oriented, and functional programming styles.
- Python प्रक्रियात्मक (procedural), ऑब्जेक्ट-ओरिएंटेड (object-oriented), और फंक्शनल प्रोग्रामिंग (functional programming) शैलियों को सपोर्ट करता है।

7. Dynamic typing:

- Variables in Python don't need to be declared with a specific type (like int, float, string); the type is determined at runtime.
- Python में वेरिएबल्स की टाइप (जैसे int, float, string) को पहले से परिभाषित करने की आवश्यकता नहीं होती, यह रनटाइम पर स्वतः निर्धारित हो जाती है।

C, Java, C# → int a=20;
Python → a=40.27; a=20;

8. Open-source and community Support:

- Python is free and open-source, with a large, supportive community for help and contributions.
- Python मुफ्त और ओपन-सोर्स है, और इसमें एक बड़ी और सहायक समुदाय है, जो मदद और योगदान के लिए हमेशा उपलब्ध रहती है।

(14) 50 %

Installing Python:

- Installing Python on your computer is quite easy. Here is a step by step explanation for Windows:
- Python को अपने कंप्यूटर पर इंस्टॉल करना काफी आसान है। इसे Windows के लिए स्टेप बाय स्टेप समझाया गया है:

1. Python वेबसाइट पर जाएँ (Go to the Python Website)

- Visit Python's official website at python.org.
- In the "Downloads" section, select the latest version for Windows.

- सबसे पहले, Python की आधिकारिक वेबसाइट python.org पर जाएँ।
- यहाँ से "Downloads" सेक्शन में जाएँ और Windows के लिए Python का लेटेस्ट वर्शन डाउनलोड करें।

2. Python इंस्टॉलर डाउनलोड करें (Download the Installer):

- Click on the download button for the latest version (e.g., "Python 3.13.0").
- This will start downloading the installer file.
- डाउनलोड पेज पर "Python X.X.X" (जैसे Python 3.13.0) के बटन पर क्लिक करें। इससे इंस्टॉलर फाइल डाउनलोड हो जाएगी।

3. इंस्टॉलर को ओपन करें (Open the Installer)

- Once the download completes, double-click the installer file to start the installation.
- डाउनलोड होने के बाद, उस इंस्टॉलर फाइल पर डबल-क्लिक करें ताकि इंस्टॉलेशन शुरू हो सके।

- **Important**: On the setup screen, check the box that says "Add Python to PATH." This allows you to access Python easily from any directory.
- **महत्वपूर्ण**: इंस्टॉलेशन स्क्रीन पर "Add Python to PATH" का विकल्प देखें और उसे चेक करें। इससे Python को आसानी से एक्सेस किया जा सकेगा।

4. इंस्टॉल पर क्लिक करें (Choose Install Options)

- You'll see options for "Install Now" and "Customize installation."
- For most users, simply clicking Install Now is the easiest choice.

- Install Now" या "Customize installation" विकल्प दिखाई देगा।
- शुरुआती उपयोगकर्ताओं के लिए "Install Now" पर क्लिक करना आसान होता है।

- If you want to install Python in a specific folder, select Customize installation and choose your preferred directory.
- यदि आप विशेष स्थान पर Python इंस्टॉल करना चाहते हैं, तो "Customize installation" का चयन कर सकते हैं और फिर अपनी पसंद की डायरेक्टरी चुन सकते हैं।

5. इंस्टॉलेशन पूरा होने का इंतजार करें(Complete the Installation)

- The installation will take a few minutes. Once it finishes, you'll see a message that says "Setup was successful."
- अब इंस्टॉलेशन कुछ मिनटों में पूरा हो जाएगा। पूरा होने के बाद, आपको "Setup was successful" का संदेश दिखाई देगा।

6. इंस्टॉलेशन की जाँच करें (Verify the Installation)

- Open the Command Prompt by typing "cmd" in the Windows search bar and pressing Enter.
- In the Command Prompt, type python --version and press Enter.
- If Python is installed correctly, it will display the version (e.g., Python 3.13.0).

- इंस्टॉलेशन चेक करने के लिए, Windows में "cmd" टाइप करें और कमांड प्रॉम्प्ट खोलें।
- फिर इसमें python --version टाइप करें और एंटर दबाएँ। यदि Python सही से इंस्टॉल हो गया है
- तो आपको Python का वर्शन (जैसे Python 3.13.0) दिखाई देगा।

CMD → Python__version
↓
Enter
↓
Current version of Python

Executing Python from the Command Line:

कमांड लाइन से पायथन का निष्पादन(execution):

1. Command Prompt खोलें (Open Command Prompt):

- Go to the Start menu, type “cmd” in the search bar, and open Command Prompt.
- Windows में Start मेन्यू में जाकर “cmd” टाइप करें और Command Prompt को ओपन करें।

2. Python को चेक करें (Check Python Installation):

- In CMD, type python --version and press Enter. If Python is installed correctly, you should see the Python version displayed.
- CMD में python --version टाइप करें और एंटर दबाएँ। अगर Python सही से इंस्टॉल हुआ है, तो Python का वर्शन दिखाई देगा।

3. Python इंटरैक्टिव मोड का उपयोग करें (Use Python Interactive Mode):

- To enter Python's interactive mode directly in CMD, simply type python and press Enter.
- You can now enter and run Python commands line by line. To exit, type exit() and press Enter.

- अगर आप सीधे Python कमांड्स रन करना चाहते हैं, तो बस python टाइप करके एंटर दबाएँ।
- अब आप Python इंटरैक्टिव मोड में हैं, और कोई भी Python कोड यहाँ चला सकते हैं। इसे छोड़ने के लिए exit() टाइप करें और एंटर दबाएँ।

CMD →

```
>>> Print(3+2)
>>> 5
```

⇒ No file save

exit

4. Python स्क्रिप्ट फाइल रन करें (Run a Python Script):

- First, create a Python script file (e.g., example.py) and save it with some code.
- In CMD, navigate to the folder where your Python file is saved. Use the cd command to change directories (e.g., cd Documents).
- To run your script, type python example.py and press Enter.

- सबसे पहले, Python में एक स्क्रिप्ट (.py) फाइल बनाएं, जैसे example.py। इस फाइल में अपना कोड लिखें और इसे सेव करें।
- अब CMD में उस फोल्डर तक जाएँ जहाँ आपकी Python फाइल सेव है। उदाहरण के लिए, अगर आपकी फाइल "Documents" फोल्डर में है, तो cd Documents टाइप करें।
- अपनी स्क्रिप्ट रन करने के लिए python example.py टाइप करें और एंटर दबाएँ।

5. परिणाम देखें(View Output):

- The output of your script will appear directly in the Command Prompt.
- With these steps, you should be able to run Python scripts easily through CMD.
- आपकी स्क्रिप्ट का आउटपुट स्क्रीन पर दिखाई देगा।
- इस प्रकार, आप CMD से Python स्क्रिप्ट आसानी से चला सकते हैं।

Integrated Development and Learning Environment (IDLE):

- Here's a step-by-step guide to running Python code in IDLE (Integrated Development and Learning Environment):
- Python को IDLE (Integrated Development and Learning Environment) में चलाना बहुत आसान है। इसे सरल स्टेप्स में समझते हैं:

1. IDLE खोलें(Open):

- Go to the Start menu and type "IDLE." Click on Python IDLE when it appears to open the interface
- सबसे पहले, Start मेन्यू में जाएँ और "IDLE" टाइप करें। जब "Python IDLE" दिखाई दे, उस पर क्लिक करें। इससे Python का IDLE इंटरफ़ेस खुल जाएगा।

2. इंटरएक्टिव मोड में कोड चलाएँ (Run Code in Interactive Mode)

- In the IDLE interface, you'll see the Python Shell. You can type commands directly here.
- For example, type print("Hello, World!") and press Enter. The output will display immediately.

- IDLE खुलने के बाद, आपको एक इंटरएक्टिव शेल (Python Shell) दिखाई देगा, जहाँ आप सीधे Python कमांड्स टाइप कर सकते हैं।
- जैसे, print("Hello, World!") टाइप करें और Enter दबाएँ। आपको तुरंत आउटपुट दिखाई देगा।

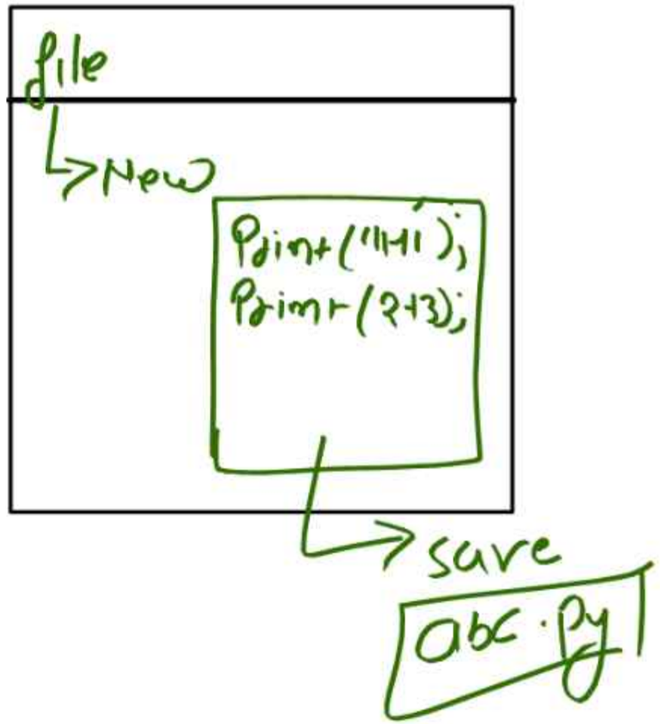
```
>>> Print("Hello")
>>> Hello
>>> Print(3+2+4)
>>> 9
```

3. एक नई स्क्रिप्ट बनाएं (Create a New Script):

- If you want to write a longer script, go to the File menu and select New File. This will open a new editor window where you can write your code.
- अगर आप बड़ी स्क्रिप्ट लिखना चाहते हैं, तो File मेन्यू में जाएँ और New File पर क्लिक करें। इससे एक नया एडिटर विंडो खुल जाएगा।
- इस एडिटर में अपना Python कोड लिखें

Example: print(" Hello");
print(" good morning Students ");

save file :- first.py



4. स्क्रिप्ट सेव करें (Save Your Script)

- After writing your code, go to File and click Save. Save your file with a .py extension, such as example.py.
- कोड लिखने के बाद, File मेन्यू में जाकर Save पर क्लिक करें और अपनी फाइल को .py एक्सटेंशन के साथ सेव करें, जैसे example.py।

5. स्क्रिप्ट रन करें (Run the Script)

F5 → Run → Run

- To run your script, go to the Run menu in the editor window and click on Run Module or press F5.
- The output will appear in the Python Shell.
- अपनी स्क्रिप्ट रन करने के लिए, एडिटर विंडो में Run मेन्यू पर जाएँ और Run Module (F5 शॉर्टकट) पर क्लिक करें।
- आउटपुट Python Shell में दिखाई देगा।

Editing Python file:

- Editing a Python file involves opening, modifying, and saving changes in a .py file to update the code.
- Python फाइल को एडिट करना यानी उसमें बदलाव करना, उसे सेव करना और फिर चेंजेस को टेस्ट करना।

Steps:-

1. Python फाइल को ढूँढें (Locate Your Python File):

- Find the .py file you want to edit on your computer. Python files have a .py extension, like example.py.
- Python फाइल को एडिट करना यानी उसमें बदलाव करना, उसे सेव करना और फिर चेंजेस को टेस्ट करना।

add.py

add.c

2. फाइल को ओपन करें (Open the File in an Editor):

- You can edit Python files in text editors like Notepad (for simple edits), or code editors like Visual Studio Code, Sublime Text, or Python IDLE for more advanced features.
- To open, right-click the file, select Open With, and choose your preferred editor.

- फाइल को एडिट करने के लिए, आप किसी भी टेक्स्ट एडिटर का उपयोग कर सकते हैं, जैसे कि Notepad, Visual Studio Code, या Python IDLE।
- फाइल पर राइट-क्लिक करें और Open With चुनें। फिर अपनी पसंद का एडिटर चुनें, जैसे IDLE या Notepad।

3. कोड में बदलाव करें (Make Changes in the Code):

- Once opened, you'll see the Python code in the editor. Make any changes you need, such as modifying variables, adding new functions, or fixing errors.

For example,

- if the code has print("Hello"),
- you can edit it to print("Hello, World!").

- एक बार जब आप इसे खोलते हैं, तो आपको एडिटर में Python कोड दिखाई देगा। आप अपनी ज़रूरत के अनुसार बदलाव कर सकते हैं — जैसे कि वेरिएबल्स को बदलना, नई फ़ंक्शन्स जोड़ना या किसी गलती (एरर) को ठीक करना।

उदाहरण के लिए,

- print("Hello") है,
- तो आप इसे print("Hello, World!") में बदल सकते हैं।

4. फाइल को सेव करें (Save the File):

- After making changes, remember to save the file to apply the edits.
- In most editors, you can go to File > Save or press Ctrl + S on your keyboard.
- बदलाव करने के बाद, File > Save पर क्लिक करें या Ctrl + S दबाएँ ताकि आपके बदलाव सेव हो जाएँ।

5. फाइल को सेव करें (Save the File):

- If you're using IDLE, go to Run > Run Module (or press F5).
- अगर आप IDLE का उपयोग कर रहे हैं, तो Run > Run Module पर क्लिक करें या F5 दबाएँ।

Python Documentation:

- Python Documentation is a comprehensive resource provided by the Python Software Foundation to help users understand and use the Python programming language.
- It contains official, detailed information on Python's syntax, libraries, functions, and modules, serving as a guide for developers at all levels.

- Python Documentation का मतलब है Python प्रोग्रामिंग भाषा के बारे में सभी जरूरी जानकारी का संग्रह। इसमें Python के सभी फंक्शन्स, लाइब्रेरीज़, सिंटैक्स, और प्रोग्रामिंग के नियमों की पूरी जानकारी होती है।
- Python Documentation प्रोग्रामर्स के लिए एक गाइड की तरह है, जो उन्हें Python के अलग-अलग फीचर्स को समझने और सही तरीके से इस्तेमाल करने में मदद करता है। आइए इसे हिंदी में सरल शब्दों में समझते हैं:

Python Documentation का उद्देश्य (Purpose of Python Documentation):

- Python Documentation is designed to support users in learning and applying Python effectively.
- Python Documentation का मुख्य उद्देश्य डेवलपर्स को Python भाषा को समझने और इस्तेमाल करने में सहायता प्रदान करना है। इसमें Python के बेसिक से लेकर एडवांस टॉपिक्स तक का विवरण होता है

1. **Python Syntax**: Rules for writing Python code.
   - Python का सिंटैक्स (कैसे कोड लिखा जाता है)

- **Built-in Functions:** Details on Python's standard functions, like print() and len().
- बिल्ट-इन फंक्शन्स (जैसे print(), len())

- **Libraries and Modules:** In-depth information on libraries such as math, datetime, and os.
- लाइब्रेरीज़ (जैसे कि math, datetime, os)
- **Data Types:** Documentation on types like int, float, string, list, and dictionary.
- डेटा टाइप्स (जैसे int, float, string, list, dictionary)
- **Classes and Objects:** Information for working with Object-Oriented Programming in Python.
- क्लासेस और ऑब्जेक्ट्स (ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग)

Contents of Python Documentation:

- **Tutorials:** A section with tutorials for beginners to get started with Python basics.
- इसमें नए उपयोगकर्ताओं के लिए एक बेसिक Python ट्यूटोरियल होता है जो उन्हें Python की बेसिक चीजों से परिचित कराता है।

- **Library Reference:** Detailed information on Python's built-in libraries, explaining functions, classes, and usage examples.
- Python के साथ आने वाली सभी इनबिल्ट लाइब्रेरीज़ का वर्णन, उनकी फंक्शनलिटी और उनका उपयोग कैसे किया जाए, यह इसमें बताया गया है।

- **Language Reference:** Technical details on Python's syntax, rules, and conventions.
- Python भाषा के सभी सिंटैक्स और नियमों की जानकारी यहाँ होती है।

() → tuple

{} → set

- **Index and Search:** A built-in index and search tool, making it easy to locate specific topics or functions
- इसमें एक इंडेक्स और सर्च फीचर होता है, जिससे यूज़र को आसानी से किसी भी टॉपिक के बारे में जानकारी मिल सकती है।

search →
- → tuple
- → list
- → set.

Getting Help:

- Here are some simple ways to get help in Python
- Python में मदद पाने के लिए कई तरीके हैं। यहाँ कुछ आसान तरीके बताए गए हैं:

1. help() function:

- Use the help() function in Python to get information about any function, module, or class. For example: help (print)
- Python में help() फ़ंक्शन का उपयोग करके किसी भी फ़ंक्शन, मॉड्यूल, या क्लास की जानकारी देख सकते हैं। उदाहरण के लिए: help(print)

help(list)

2. dir() function:

- The dir() function provides a list of all the attributes and methods available in an object. For example: dir(list)
- dir() फ़ंक्शन से किसी ऑब्जेक्ट में उपलब्ध सभी फ़ंक्शंस और एट्रीब्यूट्स की लिस्ट मिलती है। उदाहरण: dir (list)

3. Documentation Strings (Docstrings):

- Most functions and classes in Python have documentation strings, or "docstrings." When you use the help() function, it shows this information.
- Python में अधिकतर फ़ंक्शंस और क्लासेज के साथ डॉक्यूमेंटेशन स्ट्रिंग (docstrings) होते हैं। अगर आप help() फ़ंक्शन का उपयोग करते हैं, तो वह डॉक्यूमेंटेशन स्ट्रिंग को दिखाता है।

4. Documentation:

- Python's official documentation website (https://docs.python.org) provides detailed information on all modules and functions. This is very helpful when you need in-depth details on a specific topic.
- Python की ऑफ़िशियल वेबसाइट (https://docs.python.org) पर सभी मॉड्यूल्स और फ़ंक्शंस की जानकारी होती है। यह बहुत उपयोगी होता है जब आपको किसी खास टॉपिक पर गहराई से जानकारी चाहिए।

5. IDE Integrated Help:

- If you're using an IDE like PyCharm, VS Code, or others, they often have built-in help options. Just hover over a function or class, and some IDEs provide shortcuts to view documentation directly.
- अगर आप PyCharm, VS Code, या अन्य कोई आईडीई (IDE) का उपयोग कर रहे हैं, तो उनमें भी मदद का विकल्प होता है। बस फ़ंक्शन या क्लास के ऊपर होवर करें और शॉर्टकट कुंजियों का उपयोग करें।

imp

Python is Dynamically Typed:

- In Python, dynamic typing means you don't need to define the data type (like int, float, string) of a variable in advance. Python automatically detects the type of the variable based on the value assigned to it.

- Python में डायनामिक टाइपिंग का मतलब है कि आपको किसी वेरिएबल के डेटा टाइप (जैसे int, float, string) को पहले से तय नहीं करना पड़ता। Python खुद ही वेरिएबल के टाइप को समझ लेता है और उसे असाइन कर देता है।

C → lang →

int ab=20;

Char c= 'A';

a=20.24 → type → class → float

b=20 → type → integer

c="Hello" → type → string.

Reserved words (keywords):

- Reserved words (or keywords) are words that are already defined in the Python language for a specific purpose. You cannot use these words as the name of a variable, function, or anything else. These words are necessary for Python to work, and they are specially defined by the system.

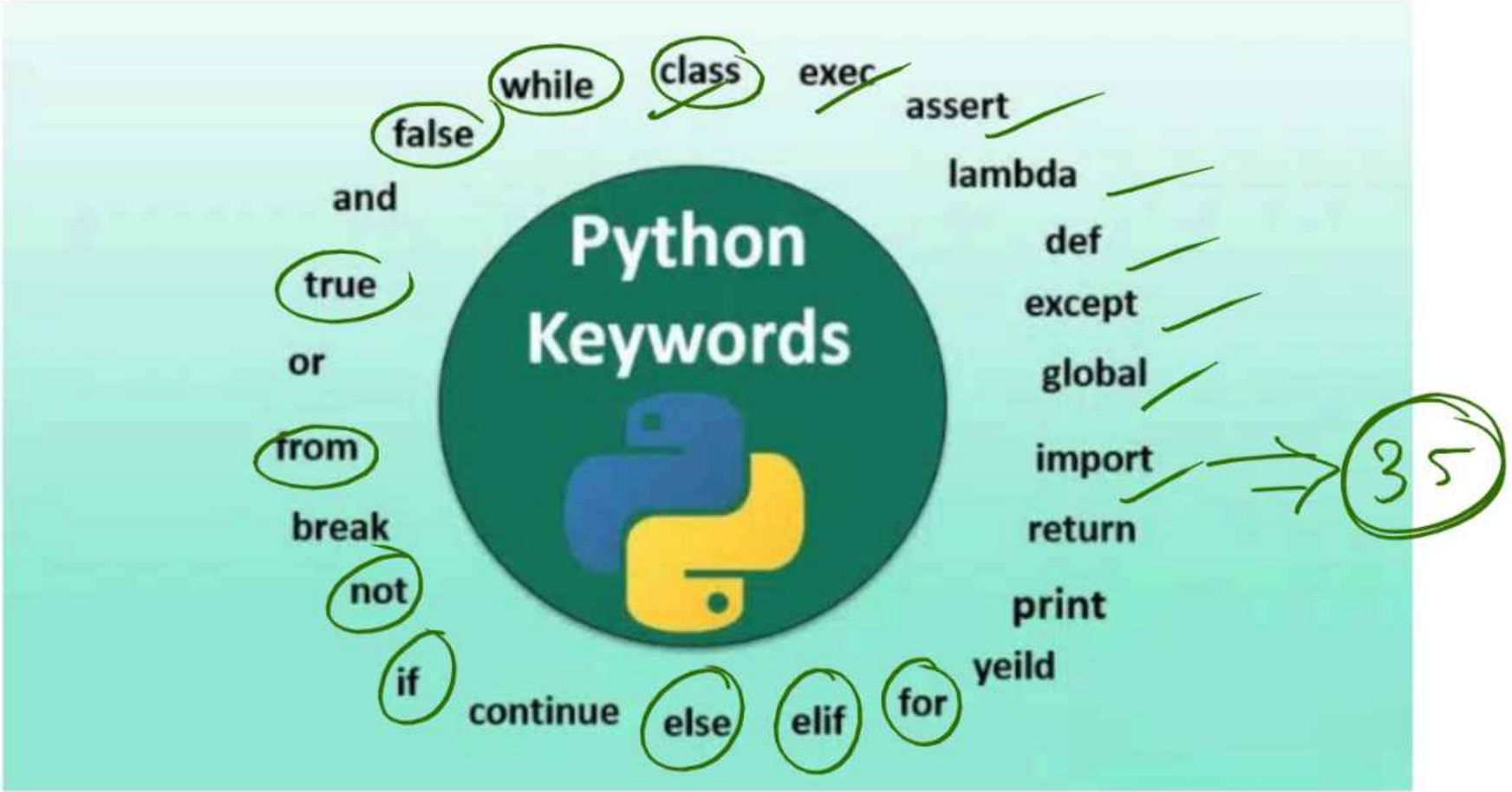
- Reserved words (या keywords) वो शब्द होते हैं जो Python भाषा में पहले से ही किसी विशेष उद्देश्य के लिए निर्धारित होते हैं। इन शब्दों का उपयोग आप किसी वेरिएबल, फंक्शन, या किसी अन्य चीज़ के नाम के रूप में नहीं कर सकते। ये शब्द Python के काम करने के लिए जरूरी होते हैं, और इन्हें सिस्टम द्वारा विशेष रूप से परिभाषित किया जाता है।

Example: if, else,for,while, def, class

C → Pro- 32
Python → 35
Java → 52

No → variable_name
→ function_name

Reserved words (keywords):



Naming Conventions:

- Following naming conventions in Python helps make the code clean, easy to understand, and consistent. It promotes good code writing habits in the programming community.
- Python में नामकरण कन्वेंशंस (Naming Conventions) का पालन करना कोड को साफ, समझने में आसान, और एकरूप (consistent) बनाने में मदद करता है। यह प्रोग्रामिंग समुदाय (community) में अच्छे कोड लेखन की आदतों को बढ़ावा देता है।

1. वेरिएबल और फ़ंक्शन के नाम(Variable and Function name):

- **Rule**: Variable and function names are written in lowercase and if the name consists of multiple words, an underscore (_) is used between those words. This is called snake_case.

akash=20; akash kumar =20 (✗)

akash_kumar=20; (✓)

- **नियम**: वेरिएबल और फ़ंक्शन के नाम को छोटे अक्षरों (lowercase) में रखा जाता है और अगर नाम में कई शब्द होते हैं, तो उन शब्दों के बीच अंडरस्कोर (_) का उपयोग किया जाता है। इसे snake_case कहते हैं।

2. क्लास के नाम(Class Name):

- **Rule**: The class name must be in CamelCase, i.e. the first letter of each word is capitalized, and the rest are lowercase. Following this rule helps distinguish classes from variables and functions.
- **नियम**: क्लास का नाम CamelCase में होना चाहिए, यानी हर शब्द का पहला अक्षर बड़े अक्षर में लिखा जाता है, और बाकी अक्षर छोटे होते हैं। इस नियम का पालन करने से क्लास को वेरिएबल्स और फ़ंक्शंस से अलग पहचाना जा सकता है।

ankit Rana → ✗

AnkitRana { }

3. कॉन्स्टेंट्स के नाम(Constant Name):

- **Rule: If you need to keep the value of a variable constant, then its name should be capitalized and use underscores between words. This naming is called UPPERCASE_SNAKE_CASE.**

- **नियम: यदि आपको किसी वेरिएबल का मान स्थिर (constant) रखना है, तो उसका नाम बड़े अक्षरों में रखना चाहिए और शब्दों के बीच अंडरस्कोर का उपयोग करना चाहिए। इस नामकरण को UPPERCASE_SNAKE_CASE कहा जाता है।**

MAX = 30;

MAX_VALUE = 40;

2 _ 1

Variable नाम नियम(Variable Name Rules):

- Start with a letter or an underscore.
- Use letters, numbers, and underscores.
- Case-sensitive.
- No Python keywords.
- No special keywords.

- अक्षर या अंडरस्कोर से शुरू करें.
- अक्षर, अंक और अंडरस्कोर का उपयोग करें.
- केस-सेंसिटिव.
- Python के कीवर्ड्स का उपयोग न करें.
- कोई विशेष कैरेक्टर नहीं.

10 → ✗

a1 → ✓

_a → ✓

(2) a123_45 (✓)

(3) A=20, a=30 → अलग-अलग

(4) if=40;

(5) → ab#34; ✗

(6) → ab_34 (✓)

UNIT 2: Syllabus

- Basic Syntax
- Comments
- String Values String Methods.
- The format Method
- String Operators.
- Numeric Data Types
- Conversion Functions
- Simple Output Simple Input
- The % Method.
- The print Function

Identifiers in python :

- Identifiers are names that we use to identify variables, functions, classes, and other elements in Python.
- You can define identifiers according to your convenience, but there are some rules for naming them:
- Identifiers वे नाम हैं जो हम Python में Variables, Functions, Classes, आदि को Unique तरीके से पहचानने के लिए देते हैं।
- Identifiers को हम अपनी सुविधा के अनुसार Define कर सकते हैं, लेकिन इनके नामकरण के कुछ नियम होते हैं:

Rules for Identifiers:

(1.) Must Start with an Alphabet or Underscore (_):

- Identifiers should start with a letter (a-z, A-Z) or an underscore (_).

(2.) Cannot Start with Numbers:

- Identifiers cannot start with a number (0-9).

(3.) No Special Characters Allowed:

- Identifiers cannot contain special characters like @, $, %, etc.

(4.) Case Sensitive:

- Python is case-sensitive, meaning name and Name would be considered two different identifiers.

Identifiers के नियम:

(1.) Alphabet और Underscore (_) से शुरू होना चाहिए:

- Identifiers को Alphabet (a-z, A-Z) या Underscore (_) से शुरू करना चाहिए।

(2) Numbers से शुरू नहीं हो सकते:

- Identifiers के नाम की शुरुआत Number (0-9) से नहीं की जा सकती।

(3)Special Characters Allowed नहीं हैं:

- Identifiers में @, $, %, जैसे Special Characters का उपयोग नहीं किया जा सकता।

(4)Case Sensitive होते हैं:

- Python में Identifiers Case Sensitive होते हैं, जैसे कि name और Name दो अलग-अलग Identifiers माने जाएंगे।

Python Statement Syntax:

- In Python, a statement is an instruction that the Python interpreter reads and executes. Statements are used to perform specific tasks, such as assigning values, printing output to the screen, checking conditions, and more.

- Python में statement एक निर्देश (instruction) होता है जिसे Python interpreter पढ़ता है और उस पर कार्य करता है। किसी कार्य को पूरा करने के लिए हम Python में अलग-अलग प्रकार के statements का उपयोग करते हैं। हर statement एक specific कार्य करता है, जैसे कि वैल्यू असाइन करना, स्क्रीन पर प्रिंट करना, किसी कंडीशन को चेक करना, आदि।

if (condition):

Print("string"):

a=20;

Types of Python Statement:

(1.) Assignment Statement

- The assignment statement is used to assign a variable to a variable.
- असाइनमेंट स्टेटमेंट का उपयोग करने के लिए किसी वेरिएबल को वैरिएबल असाइन करना होता है।

example:

Variable-Name = value

assignment-operator

a = 20 :

variable-name

value

(2.) Print Statement

- The print() statement is used to display text or values on the screen
- print() statement का उपयोग स्क्रीन पर टेक्स्ट या वैल्यू को दिखाने के लिए किया जाता है

example:

syntax:- Print Keyword ("string")

ex= Print("Hello");

(3.) Conditional Statement (if, elif, else)

- Conditional statements are used to run a part of code based on a condition.
- Conditional statements का उपयोग किसी कंडीशन के आधार पर कोड के हिस्से को चलाने के लिए किया जाता है।

example:

if → syntax :-

```
if (condition):
    Stmt;
    Stmt;
```

```
if (condition):
    elif (condition):
        Stmt;
        Stmt.
```

} → elif →

(4.) Looping Statement (for, while)

- Looping statements are used to run a task repeatedly.
- Looping statements का उपयोग किसी कार्य को बार-बार चलाने के लिए किया जाता है।

```
for a in range ( start : end : step ) :
    stmt1;
    stmt2;
```

(5.) Function Definition Statement

- Function definition statement is used to define a function.
- Function definition statement का उपयोग किसी फंक्शन को define करने के लिए किया जाता है।

```
Return_type fn_name ( arg ) :
    stmt;
    return;
```

imp

Comment in Python :

- In Python, comments are used to explain the code, and they are ignored by the computer, meaning they do not get executed. Comments are helpful for programmers to understand and document their code.
- Python में comment का उपयोग उस कोड को समझाने के लिए किया जाता है जो सिर्फ प्रोग्रामर के लिए होता है, और इसे कंप्यूटर इग्नोर कर देता है यानी रन नहीं करता।

c (a,b): # this is addition function.
return a+b;

Types of Comments in Python:

Single-line comment: For a single line comment, we use #.

- Single-line comment: एक लाइन के comment के लिए हम # का उपयोग करते हैं।

Print(3+2) → execute
Print(6+3) → 9 → o/p

Alt+3 → Comment
Alt+4 → Uncomment

Multi-line comment:

- For multiple lines, we use either ''' or """ around the text.
- Multi-line comment: कई लाइनों के comment के लिए ''' या """ का उपयोग करते हैं। इसे डबल या सिंगल कोट्स के बीच में लिखते हैं।

Example:

```
'''  or """
>>> Print(6+3+ Hello)
>>> a=20
>>> c=40
>>> Print(a+c)  ''' or """
```

more than 1-line comment → multi-line

```
>>> Print(4*3);
```

→ 12

String: (#) imp

- String in Python is a data type used to store text. It can be understood as a group of characters, such as "Hello, World!", "Python", or "12345". String in Python is written between single (' ') or double quotes (" ").

- Python में String एक डेटा टाइप है जिसका उपयोग टेक्स्ट को स्टोर करने के लिए किया जाता है। इसे कैरेक्टर्स का एक समूह समझा जा सकता है, जैसे "Hello, World!", "Python", या "12345"। String को Python में सिंगल (' ') या डबल क्वोट्स (" ") के बीच लिखा जाता है।

Note → Character_type Not allow in Python

↳ sequence of character is known as "String".

↳ single quotes (' ') } दोनो use
↳ double quotes (" ") } कर सकते हैं

a = 'A'; (✓)
a = "Hello"; ✓

$$a = \text{"SEMESTER_ADDA"}$$

S	E	M	E	S	T	E	R	_	A	D	D	A
0	1	2	3	4	5	6	7	8	9	10	11	12

String की विशेषताएँ (Characteristics of String):

1. Immutable:

- Once a String is created it cannot be changed. If we want to change a String, a new String has to be created in Python.
- बार String बनने के बाद उसमें बदलाव नहीं किया जा सकता। यदि हम String को बदलना चाहते हैं, तो Python में एक नई String बनानी होगी।

a = "Hello";

b = "Hello world";

2. Indexing and Slicing:

- Every character in a string has an index which can be accessed starting from 0. For example, in "Python", the index of P is 0, y is 1, and so on.
- String के हर कैरेक्टर का एक इंडेक्स होता है, जिसे 0 से शुरू करके एक्सेस किया जा सकता है। जैसे "Python" में P का इंडेक्स 0 है, y का 1 है, और इसी तरह आगे।

String Methods:

- Python has many methods to work with strings, which makes it easier to manipulate strings. Let's understand some important string methods:
- Python में String के साथ काम करने के लिए कई methods हैं, जिनसे String को manipulate करना आसान होता है। कुछ महत्वपूर्ण String methods के बारे में समझते हैं:

1. len()

- This method tells the length of the String, i.e. how many characters are there in the String.
- यह method String की लंबाई को बताता है, यानी String में कितने कैरेक्टर्स हैं।

```
text = "Semester Adda"
print(len(text))  # Output: 13
```

```
>>> a = "Hello"
>>> print(len(a))
>>> 5
```

2. lower()

- This method converts the String to lowercase letters.
- यह method String को छोटे अक्षरों (lowercase) में बदल देता है।

```
text = "Hello, World!"
print(text.lower())  # Output: hello, world!
```

3.upper()

- This method converts the String to uppercase letters.
- यह method String को बड़े अक्षरों (uppercase) में बदल देता है।

```
text = "Hello, World!"
print(text.upper())  # Output: HELLO, WORLD!
```

4. strip()

- This method removes the whitespace at the beginning and end of the string.
- यह method String के शुरू और अंत में whitespace (खाली जगह) को हटा देता है।

```
text = "  Hello, World!  "
print(text.strip())  # Output: "Hello, World!"
```

" Hell Hi "

5. replace()

- This method is used to replace a particular character or word in a String.
- यह method String में किसी विशेष कैरेक्टर या शब्द को बदलने के लिए उपयोग होता है।

```
text = "Hello, World!"
print(text.replace("World", "Python"))  # Output: Hello, Python!
```

Python

Past current

6. split()

- This method breaks the string into a list. By default it breaks on the basis of space.
- यह method String को एक list में तोड़ने का काम करता है। Default रूप से यह स्पेस के आधार पर तोड़ता है।

```
text = "Hello World"
print(text.split())  # Output: ['Hello', 'World']
```

7. join()

- This method is used to join all the elements of the list into a string.
- इस विधि का उपयोग list के सभी तत्वों को एक स्ट्रिंग में जोड़ने के लिए किया जाता है।

```
words = ["Hello", "World"]
print(" ".join(words))  # Output: "Hello World"
```

8. find()

- This method returns the first index of the given character or word in the string. If it is not in the string then it returns -1.
- यह method String में दिए गए कैरेक्टर या शब्द का पहला इंडेक्स बताता है। अगर वह String में नहीं है तो -1 लौटाता है।

```
text = "Hello, World!"
print(text.find("World"))  # Output: 7
```

("Python") → (-1)

H e l l o W o r l d
1 2 3 4 5 6 7 8 9 10 11

9. count()

- This method counts the number of occurrences of a particular character or word in a String.
- यह method String में किसी विशेष कैरेक्टर या शब्द की संख्या को गिनता है।

```
text = "banana"
print(text.count("a"))  # Output: 3
```

b → 1
a → 3
n → 2

10. capitalize()

- This method makes the first character of the string uppercase and the rest lowercase.
- यह method String के पहले कैरेक्टर को uppercase (बड़ा) बनाता है और बाकी को lowercase में कर देता है।

```
text = "hello, world!"
print(text.capitalize())  # Output: Hello, world!
```

a = "hello hi"
Print (a.capitalize())
⇒ (Hello Hi) → O/P

11. startswith()

- The startswith() method checks whether the String starts with a particular character or word.
- startswith() method चेक करता है कि String एक विशेष कैरेक्टर या शब्द से शुरू होती है या नहीं।

```
text = "Hello, World!"
print(text.startswith("Hello"))  # Output: True
```

12.endswith()

- The endswith() method checks whether a String ends with a particular character or word.
- endswith() method चेक करता है कि String एक विशेष कैरेक्टर या शब्द पर समाप्त होती है या नहीं।

```
text = "hello, world!"
print(text.endswith("World!"))   # Output: True
```

13 .isalpha()

- It checks whether the String contains only letters (A-Z, a-z).
- यह चेक करता है कि String में केवल अक्षर (A-Z, a-z) हैं या नहीं।

```
text = "Hello"
print(text.isalpha())  # Output: True
```

14.isdigit()

- Checks whether the String contains only numbers (0-9).
- यह चेक करता है कि String में केवल नंबर (0-9) हैं या नहीं।

```
num = "1234"
print(num.isdigit())   # Output: True
```

15 .isalnum()

- Checks whether the String contains only letters or numbers.
- यह चेक करता है कि String में केवल अक्षर या नंबर हैं या नहीं।

```
alphanumeric = "Hello123"
print(alphanumeric.isalnum())  # Output: True
```

Hello1234@ → false

The format Method:

- The format() method in Python is used to format a string. This method allows us to insert values into placeholders { } within a string.

- In the format() method, the placeholders inside { } are replaced with the value that we pass inside the format() method.

- Python में format() method का उपयोग String को format (प्रारूपित) करने के लिए किया जाता है। यह method हमें एक String के अंदर placeholders { } में वैल्यूज को इनसर्ट करने की सुविधा देता है।
- format() method में { } के अंदर जो placeholder होते हैं, वे उस वैल्यू से रिप्लेस हो जाते हैं, जिसे हम format() method के अंदर पास करते हैं।

Syntax:

```
String in {placeholder}".format(value)
```

Example:

```
name = "Vikas"
age = 20
text = "My name is {} and I am {} year old.".format(name, age)
print(text)
# Output: My name is Vikas and I am 20 year old.
```

1. Positional Arguments:

- We can also provide position in placeholders { } to specify which value should go in which placeholder.
- हम placeholders { } में position दे सकते हैं ताकि यह समझाया जा सके कि कौन सी वैल्यू किस placeholder में जानी चाहिए।

Example:

```
text = "first no{0}, second no {1}, third no {2} ".format(5, 10, 15)
print(text)
```

Output:- first no 5, second no 10, third no 15

2. Named (Keyword) Arguments:

- We can also name the placeholders to make inserting values more clear.
- हम placeholders को नाम भी दे सकते हैं ताकि वैल्यूज को और स्पष्टता के साथ इनसर्ट किया जा सके।

Example:

```
text = "My name is {name} and  I am {age} year old.format(name=”
vikas”, age= 20);
print(text)
```

Output:- My name is vikas and i am 20 year old

3. Mixed Positional And Arguments:

- You can also use positional and named arguments together in format().
- आप format() में positional और named arguments को साथ में भी उपयोग कर सकते हैं।

Example:

```
text = "My age is {0} and My name is {name}.format("20", name =
"vikas")
print(text)
```

Output- My age is 20 and My name is vikas

imP

String Operator:

- Python has a variety of operators to work with strings. These operators are used to add, repeat, and compare strings.
- Python में String के साथ काम करने के लिए कई प्रकार के ऑपरेटर्स होते हैं। इन ऑपरेटर्स का उपयोग String को जोड़ने, दोहराने और तुलना करने के लिए किया जाता है।

1. Concatenation Operator (+):

- This operator is used to join two strings.
- यह ऑपरेटर दो Strings को जोड़ने के लिए उपयोग किया जाता है।

Example:

```
str1 = "Hello"
str2 = "World"
result = str1 + " " + str2
print(resulet)
# Output: Hello World
```

```
a = "Hi"
b = "bye"
Print( a + "  " + b)
```

Hi bye → o/p

2. Repetition Operator (*):

- This operator is used to repeat a string multiple times.
- यह ऑपरेटर String को कई बार दोहराने के लिए उपयोग होता है।

Example:

```
str1 = "Hello "
result = str1 * 3
print(result)
# Output: Hello Hello Hello
```

str1 * 5

Hello → 5

Hello
Hello
Hello
Hello

3. Membership Operators (in and not in)

- The in operator checks if a substring (small part) is present in a String.
- in ऑपरेटर चेक करता है कि कोई substring (छोटा हिस्सा) String में मौजूद है या नहीं।

Example:

```
text = "Hello, World!"
print("Hello" in text)     # Output: True
```

- The not in operator checks if a substring is not present in a String.
- not in ऑपरेटर चेक करता है कि substring String में मौजूद नहीं है।

Example:

```
text = "Hello, World!"
print("Python" not in text) # Output: True
```

4. Comparison Operators (==, !=, >, <, >=, <=):

- These operators are used to compare two strings.
- ये ऑपरेटर्स दो Strings की तुलना करने के लिए उपयोग होते हैं।

(1): ==: It checks whether the two Strings are equal or not.

- ==: यह चेक करता है कि दोनों Strings बराबर हैं या नहीं।

Example:

```
str1 = "Apple"
str2 = "Banana"
print(str1 == str2)    # Output: False
```

a = "Hello"
b = "Hello"

(2): !=: It checks whether the two Strings are not equal.

- यह चेक करता है कि दोनों Strings बराबर नहीं हैं।

Example:

```
str1 = "Apple"
str2 = "Banana"
print(str1 != str2)    # Output:True
```

(3): > and <: Checks which string comes next in alphabetical order.

- यह चेक करते हैं कि कौन-सी String alphabetical order में आगे है।

Example:

```
str1 = "Apple"
str2 = "Banana"
print(str1 > str2)    # Output: False
```

Example:

```
str1 = "Apple"
str2 = "Banana"
print(str1 < str2)    # Output:True
```

5. Slice Operator ([:])

- This operator is used to extract a part of a string.
- यह ऑपरेटर String के एक हिस्से को निकालने के लिए उपयोग होता है।

Nospace

Example:

```
text = "Hello, World!"
print(text[0:5])    # Output: Hello
print(text[:5])     # Output: Hello
print(text[7:])     # Output: World!
```

[: 5]
start (0)
end
step (1)

H	e	l	l	o	,	w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

[7 :]
start

[start : end : step]

[0 : 5] → default → 1
start
end

Data Types:

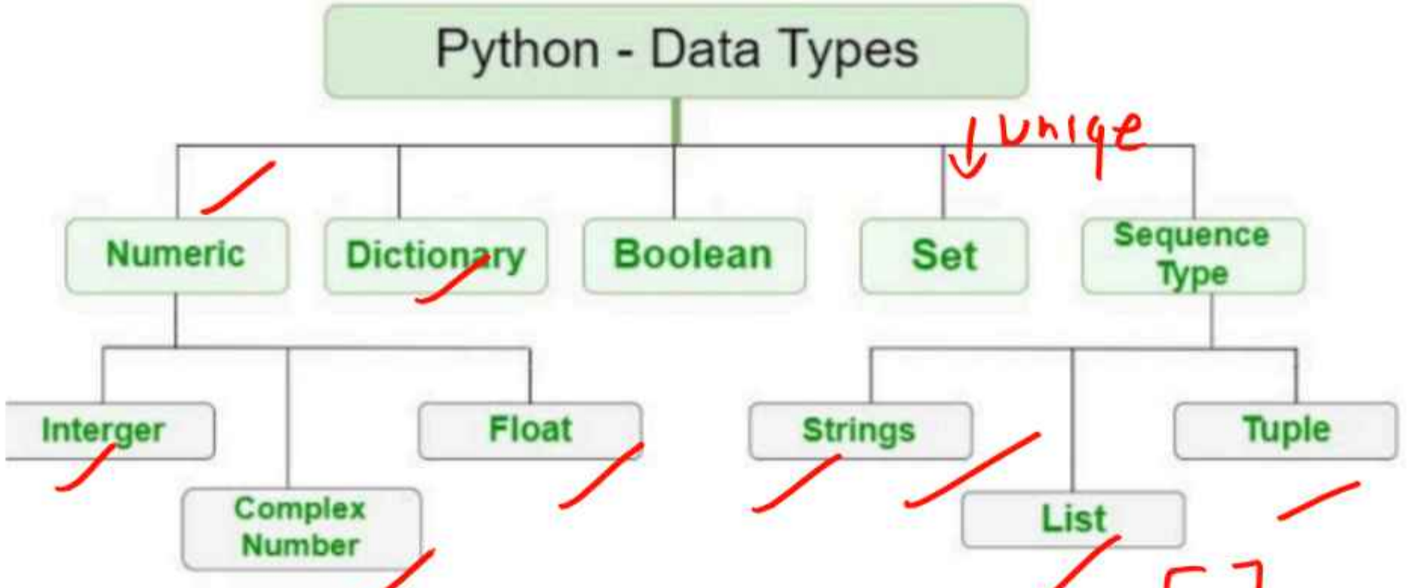
- Data Types in Python are used to determine what type of data will be stored in a variable.
- Python में डेटा टाइप्स (Data Types) का उपयोग यह निर्धारित करने के लिए किया जाता है कि किसी वैरिएबल में किस प्रकार का डेटा स्टोर होगा।

b = 20.25

a = 20

Print (type(a))
↓
int

Print (type(b))
↓
float



unique

( )

[ ]

integer a+bj → img.

Numeric Data Types:

- Numeric Data Types in Python are used to store and process numbers. There are mainly three types of numeric data types in Python: int (integer), float (decimal number), and complex (complex number).
- Python में Numeric Data Types का उपयोग संख्याओं को स्टोर और प्रोसेस करने के लिए किया जाता है। Python में मुख्य रूप से तीन प्रकार के Numeric Data Types होते हैं: int (पूर्णांक), float (दशमलव संख्या), और complex (कॉम्प्लेक्स संख्या)।

1. int (Integer - पूर्णांक):

- The int data type is used to store whole numbers, without decimals. These numbers can be positive, negative, or zero.
- int डेटा टाइप का उपयोग पूरी संख्याओं को स्टोर करने के लिए किया जाता है, जिनमें दशमलव नहीं होता। ये संख्याएँ positive, negative या zero हो सकती हैं।

```
x = 10        # Positive integer
y = -5        # Negative integer
z = 0         # Zero
print(type(x))  # Output: <class 'int'>
```

2. float (Floating Point Number - दशमलव संख्या):

- The float data type is used to store numbers that have decimal places. Float numbers can be positive, negative, or zero and have decimal places.
- float डेटा टाइप का उपयोग दशमलव वाली संख्याओं को स्टोर करने के लिए किया जाता है। Float संख्याएँ positive, negative या zero हो सकती हैं और इनमें दशमलव होता है।

```
a = 3.14      # Positive float
b = -2.5      # Negative float
c = 0.0       # Zero as float
print(type(a))  # Output: <class 'float'>
```

3. complex (Complex Number - जटिल संख्या):

- The complex data type is used to store complex numbers. Complex numbers have a real part and an imaginary part. Complex numbers are written as a + bj, where a is the real part and b is the imaginary part. j is used to represent the imaginary part in Python.

- complex डेटा टाइप का उपयोग जटिल संख्याओं को स्टोर करने के लिए किया जाता है। जटिल संख्या में एक real part और एक imaginary part होता है। Complex संख्या को a + bj के रूप में लिखा जाता है, जहाँ a real part है और b imaginary part है। Python में imaginary part को दर्शाने के लिए j का उपयोग किया जाता है।

I (Img)

```
c1 = 2 + 3j
c2 = -1 - 5j
print(type(c1))  # Output: <class 'complex'>
```

Conversion Function:

- Conversion functions in Python are used to convert data types from one format to another. This is especially useful in day-to-day programming and competitive programming.

- Python में Conversion Functions का उपयोग डेटा टाइप को एक फॉर्मेट से दूसरे फॉर्मेट में बदलने के लिए किया जाता है। यह विशेष रूप से डे-टू-डे प्रोग्रामिंग और कॉम्पिटिटिव प्रोग्रामिंग में उपयोगी है।

string → int, int → float

1. int(a, base):

- This function converts any data to an integer. If the data is a string, you can also specify the base.

- यह फंक्शन किसी भी डेटा को integer में बदलता है। यदि डेटा string में हो, तो आप बेस भी निर्दिष्ट कर सकते हैं।

8 4 2 1
1 0 1 0
8+2 ⇒ 10

Example: → Binary → 2

```
s = "1010"
c = int(s, 2)  # given digit convert into Decimal
print("convert string into Interger :", c)  # output 10
```

2. float()

- This function converts any data into a floating point number.
- यह फंक्शन किसी भी डेटा को floating point number में बदलता है।

56.0

Example:

```
s = "56"
e = float(s)
print("convert string into float :", e)
```

output: convert string into float: 56.0

8 | 56 | 0
7

Binary (0, 1) → 2
→ octal (0 – 7) → 8
Hexa → (0 → 15) → 16

3. ord()

- This function converts a character to an integer.
- यह फंक्शन किसी character को integer में बदलता है।

Example:

```
s = 'A'
c = ord(s)
print("Character Unicode:", c)
```

output: Character Unicode: 65

(ASCII) A→65, B→66, C→67
a→97, b→98, c→99
0→48
1→49

4. hex()

- This function converts an integer to a hexadecimal string.
- यह फंक्शन किसी integer को hexadecimal string में बदलता है।

```
n = 56
c = hex(n)
print("Hexadecimal: ", c) # output:  Hexadecimal :0x38
```

⇒ 16

16 | 56 | 8
3

5. oct()

- This function converts an integer to an octal string.
- यह फंक्शन किसी integer को octal string में बदलता है।

```
n = 56
c = oct(n)
print("convert into octal: ", c)
```

output :convert into octal: 0o70

6. tuple()

- This function converts data into a tuple.
- यह फंक्शन डेटा को tuple में बदलता है।

```
s = "asian"
c = tuple(s)
print("string into tuple: ", c)
```

output: string into tuple: ('a', 's', 'i', 'a', 'n')

7. set()

- This function converts data into a set.
- यह फंक्शन डेटा को set में बदलता है।

```
s = "banana"
c = set(s)
print("string into set:", c)
```

output: string into set: {'b', 'a', 'n'}

b → 1
n → 2
a → 3

{b, n, a}

8. list()

- This function converts the data into a list.
- यह फंक्शन डेटा को list में बदलता है।

[ ] → typle ( )

```
s = "hello"
c = list(s)
print(" string into List:  ", c)
```

output :string into List: ['h', 'e', 'l', 'l', 'o']

9. dict():

- This function converts a tuple of key-value pairs into a dictionary.
- यह फंक्शन tuple of key-value pairs को dictionary में बदलता है।

```
tup = (('a', 1), ('b', 2))
c = dict(tup)
print("Dictionary:", c)
```

output: Dictionary : {'a': 1, 'b': 2}

19. complex(real, imag):

- This function creates a complex number from the real and imaginary parts.
- यह फंक्शन real और imaginary पार्ट से एक complex number बनाता है।

Example :

```
a = 1,b = 2
c = complex(a, b)
print("Complex ", c)
```

output: Complex नंबर: (1+2j)

Input Output function in Python:

- The input and output functions in Python are used to take data from the user and show it on the display.

- Python में input और output functions का उपयोग user से डेटा लेने और display पर दिखाने के लिए किया जाता है।

a=input("Enter the Number);

a= int(input(Enter the Number));

1. Input Function:

- In Python, the input() function is used to take data from the user.
- It always takes data as a string (text).
- If a number is to be taken, it has to be converted first.

- Python में input() function का उपयोग user से डेटा लेने के लिए किया जाता है।
- यह हमेशा डेटा को string (पाठ) के रूप में लेता है।
- यदि संख्या (number) लेनी हो, तो उसे पहले convert करना पड़ता है।

Syntax: Variabl_Name= input("enter the statement")

Example 1:

```
Name= input("Enter the Name")
print( "Your Name is: ", Name)
```

output:- Your Name is:

Example 2:

```
Number =input("Enter the Number ")
print( "Your Number  is: ", Number)
```

output:- Your Number is: string

print(type(Number))
↓
string=

2. Output Function:

- The print() function in Python is used to display something on the screen.
- It helps to output text, variables, or calculations.
- Python में print() function का उपयोग स्क्रीन पर कुछ दिखाने के लिए किया जाता है।
- यह text, variables, या calculations को output करने में मदद करता है।

Syntax: print("write simple statements ")

```
>>> Print("Hello student")
>>> Hello student
>>> Print(9+3)
>>> 12
```

The % Method:

- The % method in Python (called string formatting) is used to insert variables into strings using placeholders. This is an old-fashioned string formatting method that is still useful and easy to use.
- Python में % Method (जिसे string formatting कहा जाता है) का उपयोग strings में variables को placeholders की मदद से insert करने के लिए किया जाता है। यह पुराने तरीके का string formatting method है, जो अभी भी उपयोगी और आसान है।

Syntax: "string % values"

1. %s - String Formatting:

- %s is used to format strings.
- %s का उपयोग strings को format करने के लिए किया जाता है।

Example:

```
first_name = "Semester "
last_name = "Adda"
print("My Channel first name is %s and last name is %s" % (first_name, last_name))
```

Output: My channel first name is Semester and last name is Adda

2. %d - Signed Decimal Integer

- %d is used to format integers (whole numbers).
- %d का उपयोग integers (whole numbers) को format करने के लिए किया जाता है।

Example:

```
age = 25
print("I am %d years old" % age)
```

Placeholder

Output: I am 25 years old

3. %f - Floating Point Numbers

- %f is used to format floating-point numbers.
- You can also specify how many digits to display after the decimal point.
- %f का उपयोग floating-point numbers को format करने के लिए किया जाता है।
- आप दशमलव के बाद कितने अंकों तक दिखाना है, यह भी तय कर सकते हैं।

Example:

```
pi = 3.14159
print("Value of pi  is %.2f" % pi)  # 2 decimal places
```

कितनी value

Output: Value of pi is 3.14

4. %x और %X - Hexadecimal Integer

- In Python, the %x and %X placeholders are used to represent integers in hexadecimal format.
- Python में %x और %X placeholders का उपयोग integers (पूरी संख्याओं) को hexadecimal format में दिखाने के लिए किया जाता है।
- %x: Lowercase hexadecimal
- %X: Uppercase hexadecimal

Example:

```
number = 255
print("Hexadecimal (lowercase): %x" % number)
print("Hexadecimal (uppercase): %X" % number)
```

Output:

```
Hexadecimal (lowercase): ff
Hexadecimal (uppercase): FF
```

⇒ 255 → FF

5. %e और %E - Exponential Notation:

- In Python, the %e and %E placeholders are used to format numbers in exponential (scientific) notation.
- Python में %e और %E placeholders का उपयोग numbers को exponential (scientific) notation में format करने के लिए किया जाता है।
- %e: Lowercase hexadecimal
- %E: Uppercase hexadecimal

Example:

```
value = 123456.789
print("Exponential (lowercase): %e" % value)
print("Exponential (uppercase): %E" % value)
```

Output:

```
Exponential (lowercase): 1.234568e+05
Exponential (uppercase): 1.234568E+05
```

Chapter No-3 (languages Component)

Syllabus:

- Indenting Requirements.
- The if Statement.
- Relational and Logical Operators.
- Bit Wise Operators.
- The while Loop.
- break and continue.
- The for Loop.

1-hour

10 marks

Indenting Requirements in Python:

- Indentation is very important in Python because it determines code structure and execution.
- One of the main reasons why Python is different from other programming languages is that braces ({}) or keywords (such as begin and end) are used instead of indentation.
- Python में इंडेंटेशन बहुत महत्वपूर्ण है क्योंकि यह कोड संरचना और निष्पादन को निर्धारित करता है।
- Python की अन्य प्रोग्रामिंग भाषाओं में अलग-अलग होने का एक मुख्य कारण यह है कि इंडेंटेशन के बजाय ब्रेसिज़ ({}) या कीवर्ड (जैसे शुरू और अंत) का उपयोग किया जाता है।

C-lang → for (int i=0; i<=5; i++){
Stmt1;
Stmt2;
}

for i in range( ):
Print(i);
i++
indentation

Rules of Indentation:

1. **Minimum one indentation:**
   - Code block must be indented at least once.
   - Code block को कम से कम एक बार indent करना जरूरी है।

2. **Consistent indentation:**
   - Same spacing or tabbing should be used throughout the code.
   - पूरे code में एक जैसा spacing या tabbing उपयोग करना चाहिए।

3. **IndentationError:**
   - If indentation is not correct, Python will give IndentationError.
   - यदि indentation सही नहीं है, तो Python IndentationError देगा।

Example1:

```
for i in range(2):
    print("Tab Indent")  # Tab
    print("Space Indent")  # Space
```

error

Example2:

```
for i in range(2):
    print("Only Space Indent")  # Use only spaces or only tabs
```

tab

for i in range(2)
print(i)
print(i)

if Statement: (Conditional stmt)

- In Python, the if statement is used for decision-making. This means that the program checks a condition, and if that condition is true, it executes a specific block of code. If the condition is false, that block is skipped.
- Python में if statement का उपयोग decision-making के लिए किया जाता है। इसका मतलब है कि प्रोग्राम किसी condition (शर्त) को check करता है, और अगर वह condition सही (True) होती है, तो एक specific block का code execute करता है। अगर condition गलत (False) होती है, तो वह block skip कर दिया जाता है।

Syntax:

```
if condition:
    # Code to execute if the condition is True
```

if: This is the keyword that tells that we are checking a condition.

- यह keyword है जो बताता है कि हम एक condition check कर रहे हैं।

condition: It is an expression that evaluates to True or False.

- condition: यह एक expression होता है जो True या False में evaluate होता है।

Colon (:) : It shows the start of the block. { → Python :

- यह block की शुरुआत को दिखाता है।

Indented code:

- The code which is indented (one level inside) will be executed only if the condition is True.
- जो code indented होता है (एक level अंदर होता है), वही condition के True होने पर execute होगा।

(1) if Statement:

- first checks the condition of if.
- If the condition is True, then the code of the if block will be executed.
- If the condition is False, then Python ignores this block and executes the further code.

- Python पहले if की condition को check करता है।
- अगर condition True होती है, तो if block का code execute होगा।
- अगर condition False होती है, तो Python इस block को ignore कर देता है और आगे के code को execute करता है।

Example 1:

```
x = 10
if x > 5:  # Check if x is greater than 5
    print("x is greater than 5")
```

10>5

Output:- x is greater than 5

Example 2:

x>5

2>5 → false

```
x = 2

if x > 5:  # Check if x is greater than 5
    print("x is greater than 5")
```

x

exit

Output:-# x is lesser than 5 so no output is printed

(2) if Statement के साथ else:

- If the condition becomes False, you can use else block. This block is executed when the if condition is false.
- अगर condition False हो जाती है, तो आप else block का उपयोग कर सकते हैं। यह block तब execute होता है जब if की condition गलत हो।

a=10
b=20

(a>b)→
→true (if-block)
→false→(else-block)

Syntax:

```
if condition:
    # Code to execute if the condition is True
else:
    # Code to execute if the condition is False
```

Example:

```
x = 2
```

10

```
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

Output: x is not greater than 5

x > 5

2 > 5 → false

10 > 5 →

→ true

x is greater than 5

(3) if, elif, and , else:

- When you have multiple conditions, you can use elif (else if). This checks each condition sequentially.
- जब आपके पास multiple conditions हों, तो आप elif (else if) का उपयोग कर सकते हैं। यह sequentially हर condition को check करता है।

Syntax:

```
if condition1:
    # Code if condition1 is True
elif condition2:
    # Code if condition2 irs True
else:
    # Code if none of the conditions are True
```

else if
||
elif -> Python

Example:

```
x = 10

if x < 5:
    print("x is less than 5")
elif x == 10:
    print("x is equal to 10")
else:
    print("x is greater than 5 but not equal to 10")
```

① $x < 5$
$10 < 5 \rightarrow$ false

② $x == 10$
$10 == 10$

Output: x is equal to 10

Nested if Statement:

- Another if statement can be written inside an if statement. This is called nested if.
- if statement के अंदर दूसरा if statement लिखा जा सकता है। इसे nested if कहते हैं।

Example :

```
x = 15
if x > 10:
    print("x is greater than 10")
    if x % 2 == 0:
        print("x is even")
    else:
        print("x is odd")
```

WAP to Print even or odd No.

Output :

x is greater than 10
x is odd

12 % 2 == 0
12 % 2 == 0 → true
n is even

n % 2 == 0
15 % 2 ≠ 0 ⇒ 1

Operators: 701.

- Operators are a special type of symbol, by which we perform arithmetic and logical calculations. The thing on which the operator is operated is called operand.
- Operators एक विशेष प्रकार का चिन्ह (Symbol) होता है, जिससे हम अंकगणितीय तथा लॉजिकल संगणना करते हैं। आपरेटर को जिस पर आपरेट किया जाता है, उसे हम आपरेन्ड (operand) कहते हैं

  - Arithmetic Operators
  - Assignment Operators
  - Bitwise Operators
  - Identify Operators
  - Relational Operators/Comparision Operators
  - Logical Operators
  - Membership Operators

a + b → operand
+ → operator

a * b
* → operator

a == b
== → operator

Arithmetic Operators:

- Arithmetic operators in Python are used for mathematical calculations. These operators work on two or more values and provide results.
- Python में Arithmetic Operators का उपयोग गणितीय गणनाओं के लिए किया जाता है। ये ऑपरेटर्स दो या अधिक मानों पर काम करते हैं और परिणाम प्रदान करते हैं।

Arithmetic Operators:

OPERATOR	MEANING	EXAMPLE
+	Add	3+2 ⇒ 5
-	Subtract	3 − 2 = 1
*	Multiple	3×2 = 6
/	Division	7/2 ⇒ 3.5
//	Floor Division	7//2 ⇒ 3
**	Expone	10**2 or $(10)^2$ ⇒ 10×10=100
%	Module	12 % 5 (Reminder) ⇒ op → 2

Relational Operators:

- Relational operators are used to compare the relationship between two values or variables.
- Their result is always True or False.

- Relational operators का उपयोग दो values या variables के बीच relationship को compare करने के लिए किया जाता है।
- इनका result हमेशा True या False होता है।

Relational Operators:

OPERATOR	MEANING	EXAMPLE
<	Less than	3<5 → T
>	Greater than	8>5 → T
<=	Less than equal to	5<=10 → T
>=	Greater than equal to	10>=5 → T
==	Equal to	10==10 → T
!=	Not equal to	10!= 5 → T

Example: Relational Operators

```
a = 10
b = 5
print(a > b)   # True
print(a < b)   # False
print(a >= 10) # True
print(a <= b)  # False
print(a == 10) # True
print(a != b)  # True
```

10 > 5
10 < 5

Output:

```
True
False
True
False
True
True
```

Logical Operators:

- Logical operators are used to combine conditions.
- These operators check multiple conditions simultaneously.

- Logical operators का उपयोग conditions को combine करने के लिए किया जाता है।
- ये operators multiple conditions को एक साथ check करते हैं।

① && AND → all condition are true (True)
↳ false

② || OR → at least one condition is true (true)
else → false

③ ! Not
↳ true → false
↳ false → true.

Logical Operators:

OPERATOR	MEANING	EXAMPLE
and	Returns True if both conditions are True	(5 > 3) and (8 > 6) → True (5>3) and (8<6) → False.
or	Returns True if any one condition is True	(5 > 3) or (8 < 6) → True
not	Reverses the Boolean value	not (5 > 3) → False

(1.)and Operator:

- It will be True only if all the conditions are True.
- यह तभी True होगा जब सभी conditions True हों।

```
print((10 > 5) and (20 > 15))  # True
print((10 > 5) and (20 < 15))  # False
```

(2.) or Operator:

- It will be True only if at least one condition is True.
- यह तभी True होगा जब कम से कम एक condition True हो।

```
print((10 > 5) or (20 < 15))  # True
print((10 < 5) or (20 < 15))  # False
```

3. not Operator:

- It reverses the Boolean value.
- यह Boolean value को उल्टा कर देता है।

```
print(not (10 > 5))  # False
print(not (10 < 5))  # True
```

Bitwise Operator :

- Bitwise operators are used to perform operations at the binary level in Python. This means that these operators work by converting data into bits.
- Bitwise operators are mainly applied to numbers and are based on the binary number system.

bit- level → 0, 1

- बिटवाइज ऑपरेटर (Bitwise Operators) का उपयोग Python में बाइनरी (binary) स्तर पर ऑपरेशन करने के लिए किया जाता है। इसका मतलब है कि ये ऑपरेटर डाटा को बिट्स (bits) में बदलकर काम करते हैं।
- बिटवाइज ऑपरेटर मुख्य रूप से संख्या (numbers) पर लागू होते हैं और बाइनरी संख्या प्रणाली पर आधारित होते हैं।

1 1 0
8 4 2 1
(1 0 1)

Bitwise Operators:

OPERATOR	Example	Output
&. (AND)	5&3	1 (5 → 101, 3 → 011, 001) both high → 1, ↳ 0
I (OR)	5 I 3	7 (101, 011, 111 → 7)
^ (XOR)	5^3	6 (101, 011, 110) → both are 0 or 1 → 0
~ (NOT)	5	-6 (~7 ⇒ (−8))
>>Right shift	a=1010 (10) 5 → 2	a>>2. Output (00000010)
<< left Shift	a=1010 (10×2 = 20×2 = 40)	a<<2 output (00101000)

Membership Operator

- The second special type of operator in Python is the membership operator. It has two operators in and not in; which are used to check whether a value or variable is in a sequence or not. This sequence can be a string list, tuple or dictionary

- Python में जो दूसरा विशिष्ट प्रकार का आपरेटर है वह membership operator है। इसमें दो आपरेटर है in एवं wt in; जो यह चेक करने के लिए प्रयोग किए जाते हैं कि कोई value या variable एक sequence में है या नहीं। यह sequence कोई string list, tuple या dictionary हो सकता है

Membership Operators:

OPERATOR	Example	Output
in	a="Hello" print("H" in a)	True
not in	a="Hello" print("y" not in a)	True

The while Loop:

- The while loop in Python is used when we want to do something repeatedly as long as a condition is true.
- Python में while loop का उपयोग तब किया जाता है जब हमें किसी काम को बार-बार तब तक करना हो जब तक कोई शर्त (condition) सही (True) हो।

Working of While loop:

- The condition is checked.
- If the condition is True, the code inside the loop is executed.
- After the code is executed, the condition is checked again.
- This process continues until the condition becomes False.

- condition को चेक किया जाता है।
- अगर condition True है, तो loop के अंदर का कोड execute होता है।
- कोड execute होने के बाद, condition फिर से चेक की जाती है।
- यह प्रक्रिया तब तक चलती रहती है जब तक condition False न हो जाए।

Syntax:

```
while condition:
    # Code to execute
```

Example:

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

exit

6
5
4
~~3~~
~~2~~
~~1~~
i

1
2
3
4
5

i<=5
1<=5
2<=5
3<=5
4<=5
5<=5
6<=5

(infinite) loop:

- If you keep the condition true and never make it false, the loop will run forever.

- अगर आप condition को सही (True) ही रखते हैं और उसे कभी false नहीं करते, तो loop हमेशा चलता रहेगा।

Example:

```
i=1
while i:
    print("This is infinite loop ")
```

For loop:

- The for loop in Python is used to access each item of a sequence (such as a list, string, or range) one by one. It is very common to use it in conjunction with the range() function.
- Python में for loop का उपयोग एक sequence (जैसे list, string, या range) के प्रत्येक item को एक-एक करके एक्सेस करने के लिए किया जाता है। इसे range() function के साथ मिलाकर उपयोग करना बहुत आम है।

Syntax:

```
for variabel in range(start, stop, inc/dec)
```

Condition

Range function()

- It creates a special type of sequence containing numbers.
- It is very easy to use with a for loop.

- start: start of counting (default: 0)
- stop: where to count (exclusive i.e. this number is not included).
- step: how many increments/decrements to do in each iteration (default: 1).

- यह loop किसी sequence (list, tuple, string आदि) के हर element पर चलता है।
- इसे बार-बार दोहराए जाने वाले कार्यों के लिए उपयोग किया जाता है।

- start: गिनती की शुरुआत (default: 0)
- stop: जहाँ तक गिनना है (exclusive यानी यह number शामिल नहीं होता)।
- step: प्रत्येक iteration में कितने increment/decrement हों (default: 1)।

Example 1:

```
for i in range(5):
    print(i)
```

Output:

0
1
2
3
4

Example 2:

```
for i in range(1, 6):
    print(i)
```

Output:

1
2
3
4
5

Control statement in python :

- Control statements in Python are used to control the flow of the program. These statements decide which code will be executed when and how many times.
- Python में Control Statements का उपयोग प्रोग्राम के प्रवाह (flow) को नियंत्रित करने के लिए किया जाता है। ये statements यह तय करते हैं कि कौन सा कोड कब और कितनी बार execute होगा।

Types of Control Statements in Python

1. break
2. continue
3. pass

```
for a in range(10)
    if a==5:
        break
    print(i)
exit
```

a

0
1
2
3
4

5==5

break Statement:

- It is used to stop the loop immediately.
- As soon as the break statement is executed, the loop ends, and the control comes out of the loop.
- यह लूप को तुरंत रोकने के लिए उपयोग किया जाता है।
- जैसे ही break स्टेटमेंट execute होता है, लूप समाप्त हो जाता है, और control लूप के बाहर आ जाता है।

Example:

```
for i in range(1, 10):
    if i == 5:
        break
    print(i)
```

Output :

```
1
2
3
4
```

continue Statement:

- It skips the current iteration of the loop and moves on to the next cycle of the loop.
- The code after continue is not executed.
- यह लूप के वर्तमान iteration को स्किप करता है और लूप के अगले चक्र पर जाता है।
- continue के बाद का कोड execute नहीं होता।

Example:

```
for i in range(1, 10):
    if i == 5:
        continue
    print(i)
```

Output :

```
1
2
3
4
6
7
8
9
```

5 skip

pass Statement:

- This is a null statement.
- It is used where code is required but you don't want to write any code.
- It makes the program error-free.

- यह एक null statement है।
- इसका उपयोग उस जगह पर किया जाता है, जहाँ कोड की आवश्यकता है लेकिन आप कोई कोड लिखना नहीं चाहते।
- यह प्रोग्राम को error-free बनाता है।

Example:

```
for i in range(1, 6):
    if i == 3:
        pass
    else:
        print(i)
```

Output :

```
1
2
4
5
```

3 skip

Chapter No-4 (collections)

Syllabus:

- Introduction, Lists, Tuples, Sets, Dictionaries Sorting Dictionaries, Copying Collections.

a = 20

⇒ 10 marks

Built-in Collection Data Types:

- Python offers several built-in collection data types, each with its own characteristics and use cases:
- Python कई इन-बिल्ट कलेक्शन डेटा टाइप्स प्रदान करता है, जिनमें से प्रत्येक की अपनी विशेषताएँ और उपयोग के मामले हैं:

List:

- Ordered and mutable (changeable), Can store elements of different data types, Accessed using index values.
- क्रमबद्ध (Ordered) और परिवर्तनीय (Mutable) होती है।
- विभिन्न डेटा टाइप्स के तत्वों को स्टोर कर सकती है।
- तत्वों को इंडेक्स मान (Index Values) का उपयोग करके एक्सेस किया जाता है।

Example: my_list = [1, 2, "hello", 3.14]

int String float

Tuple:

- Ordered and immutable (unchangeable), Can contain elements of different data types, and Enclosed in parentheses ().

- क्रमबद्ध (Ordered) और अपरिवर्तनीय (Immutable) होती है।
- विभिन्न डेटा टाइप्स के तत्वों को स्टोर कर सकती है।
- कोष्ठकों () में बंद होती है।

[ ] → list

( ) → tuple

Example: my_tuple = (1, "world", 2.718)

Set:

- Unordered and mutable, Contains only unique elements, and Enclosed in curly braces {}.
- अव्यवस्थित (Unordered) और परिवर्तनीय (Mutable) होती है।
- इसमें केवल अद्वितीय (Unique) तत्व होते हैं।
- आंकार {} में बंद होती है।

{1, 1, 2, 2, 3} ✗

⇓

{1, 2, 3} ✓

Example: my_set = {1, 3, 2}

Dictionary:

- Unordered (ordered in Python 3.7+) and mutable.
- Stores data in key-value pairs.
- Enclosed in curly braces {}, with keys and values separated by colons :

- अव्यवस्थित (Python 3.7+ में क्रमबद्ध) और परिवर्तनीय (Mutable) होती है।
- डेटा को कुंजी-मूल्य जोड़ियों (key-value pairs) में स्टोर करती है।
- आंकार {} में बंद होती है, जिसमें कुंजी और मूल्य को कोलन : द्वारा अलग किया जाता है।

Example: my_dict = {"name": "Alice", "age": 30}

Key Value Key Value

List in Python:

- List is a collection of data that is ordered and changeable. It is a data type that is very versatile for Python's compound. In a list, we separate the items with commas (,) and put all the items of the entire list within square brackets [].

- List एक डाटा का कलेक्शन है, जो कि ordered तथा changeable होता है। यह एक ऐसा data types है जो कि बहुत ही versatile है python's compound के लिए। एक लिस्ट में हम आइटम्स (items) को कामा (,) से अलग-अलग करते है तथा पूरी लिस्ट के सभी items को square bracket[] में रखते हैं।

[-5] [-4] [-3] [-2] [-1] → index

a = [1 , 2 , 'A' , 'B' , 20.25]

[0] [1] [2] [3] [4] → index

List Methods:

(1): append()

- Adds an element at the end of the list
- List के अंत में एक तत्व जोड़ता है
- Example:

```
>>> a = [1, 2, 3]
>>> a.append (6)
>>> Print (a)
>>> [ 1, 2, 3, 6]
```

(2): clear()

- Removes all the elements from the list
- List से सभी तत्वों को हटाता है
- Example:

```
>>> a = [1, 2, 3]
>>> a.clear()
>>> Print(a)
```

=> O/P => [ ]

(3): Copy()

- Returns a copy of the list
- List की एक copy लौटाता है

- Example:

```
>>> a = [1, 2, 3]
>>> b = a.copy()
>>> Print(b)
>>> [1, 2, 3]
```

(4): Count()

- Returns the number of elements with the specified value
- specified value वाले element की संख्या लौटाता है

- Example:

```
>>> a=[1, 2, 3, 1, 1, 1, 1, 2]
>>> Print(a.count(1))
>>> 5
```

(5): extend()

- Add the elements of a list to the end of the current list
- किसी list के तत्वों को वर्तमान सूची के अंत में जोड़ें
- Example:

```
>>> a=[1,2,3]
>>> a.extend([4,5,6])
>>> Print(a)
>>> [1,2,3,4,5,6]
```

(6): index()

- Returns the index of the first element with the specified value
- Specified मान के साथ पहले तत्व का index लौटाता है
- Example:

```
>>> a=[1,2,3,4,5]
>>> Print(a.index(4))
>>> 3
```

[1, 2, 3, 4, 5]
(0) (1) (2) (3) (4)

(7): insert()

- Adds an element at the specified position
- specified स्थान पर एक तत्व जोड़ता है
- Example:

```
a = [1, 2, 3]
>>> a.insert(0, 4)
>>> print(a)
```

index → 0, value → 4

[4, 1, 2, 3]
(0) (1) (2) (3)

(8): pop()

- Removes the element at the specified position
- Specified स्थान पर तत्व को हटाता है
- Example:

```
a = [1, 2, 3, 4]
>>> a.pop(1)
>>> print(a)
```

(0) (1) (2) (3)

⇒ [1, 3, 4]

Tuple:

- Tuple is an immutable list. Once a Tuple is created, no changes can be made to it. Tuple is also defined in the same way as a list. Only in this, instead of square position brackets [], parenthesis () is used to separate the elements.

- Tuple एक immutable लिस्ट है। एक बार Tuple क्रिएट हो जाने पर, उसमें किसी भी प्रकार का परिवर्तन नहीं किया जा सकता है। Tuple को भी लिस्ट की तरह ही डिफाइन किया जाता है। इसमें केवल एलमेंट्स को square sition brackets [] के स्थान पर, parenthesis() का प्रयोग किया जाता है।

Example:

```
-6   -5   -4   -3   -2   -1
 0    1    2    3    4    5    6
a = ( 1, 2, 3, "Hello", "Hi", 20.25, 40.25 )
```

integer

string

float

in method:

- The in operator checks whether an element is present in a list, tuple, or string. It returns a Boolean (True/False) result.
- in operator यह चेक करता है कि कोई element list, tuple, या string में मौजूद है या नहीं। यह Boolean (True/False) result देता है।

Syntax: element in sequence

```
>>>a=( 1, 2, 3, 4, "Ram" )
>>> Print( 2 in a )
>>> true
>>> Print ("Rama" in a )
>>> false.
```

index() Method:

- The index() method is used to return the first position (index) of a given element.
- index() method किसी दिए गए element का पहला स्थान (index) बताने के लिए उपयोग किया जाता है।

Syntax: sequence.index(element, start, end)

```
        0 1 2
>>> a = ( 1, 2, 3, 4, 5 )
>>> Print(a.index(3))
>>> 2
```

count() Method:

- The count() method is used to find out how many times a specific element occurs in a tuple.
- count() method का उपयोग यह पता लगाने के लिए किया जाता है कि कोई specific element tuple में कितनी बार आता है।

Syntax: tuple.count(element)

```
>>> a = (" 1, 2, 3, 'vikas', "vikas")
>>> Print( a.count("vikas))
```

t-Name → a, method → count, element → "vikas

```
>>> 2
```

List	Tuple
• Ordered Collection	• ~~Unordered~~ Ordered Collection
• consume more memory	• consume less memory
• Mutable	• Immutable
• Dynamic	• Static
• Enclosed with []	• Enclosed with ()
• More Operations	• Less Operations
• Slow	• Fast

Set:

- A SET is an unordered collection of items in which each element is unique, there are no duplicate elements and it is immutable i.e. it cannot be changed.
- Sets are neither mappings nor sequences.
- However, the set itself is mutable i.e. we can add or remove anything.
- We use Sets for mathematical set operations like Union (|), Insertion (&), Symmetric difference (^) etc.

- SET एक items का unordered collection होता है, जिसमें प्रत्येक element यूनिक होता है, कोई भी डुप्लीकेट element नहीं होता है तथा [redacted]mutable होता है, अर्थात् इसमें परिवर्तन नहीं किया जा सकता है।
- Sets न तो mapping है और न ही sequences होता है।
- यद्यपि, the set स्वयं में mutable होता है, जिसमें हम कुछ भी add या remove कर सकते हैं।
- Sets का प्रयोग हम mathematical set operations जैसे Union (|), Insertion (&), Symmetric difference (^) आदि के लिए करते हैं।

Set Method:

1. add()

- Function: Adds a new element to the set.
- Set में एक नया element जोड़ता है।

```
>>> a={a, b, 1, 2, }
>>> Print(a)
{a, b, 1, 2}
```

Syntax : set.add(element)

```
>>> a.add(4)
>>> Print(a)
```
⇒ {a, b, 1, 2, 4}

2. update():

- Set में एक से अधिक elements जोड़ने के लिए उपयोग होता है।
- Function: Used to add more than one element to a set.

Syntax : set.update([iterable])

```
>>> a={1,2,3}
>>> a.update([6,7,8])
>>> Print(a)
```

O/P → {1, 2, 3, 6, 7, 8}

3. remove()

- The remove() method in Python is used to remove a particular element from a set. But if that element does not exist in the set, then this method throws KeyError.
- Python में remove() method का उपयोग set से किसी विशेष element को हटाने के लिए किया जाता है। लेकिन अगर वह element set में मौजूद नहीं है, तो यह method KeyError फेंकता है।

Syntax : set.remove(element)

```
>>> a={1, 2, 3, 4}
>>> a.remove(4)
>>> print(a)
>>> {1,2,3}
```

```
>>> a.remove(4)
>>> KeyError
```

4. discard()

- **Function: Removes a particular element from the set. Does not give an error if the element does not exist.**
- **Set से किसी विशेष element को हटाता है। अगर element मौजूद न हो, तो error नहीं देता।**

Syntax : set.discard(element))

```
>>> a={1,2,3,4}
>>> a.discard(5)
>>> print(a)
>>> {1,2,3,4}
```

5. pop()

- **Function: Removes a random element from the set and returns it.**
- **Set से कोई random element हटाता है और उसे return करता है।**

Syntax : set.pop()

```
>>> a={1,2,3,4}
>>> a.pop()
>>> print(a)   => {2,3,4}
```

6. clear()

- **Function: Completely clears the Set.**
- **Set को पूरी तरह से खाली कर देता है।**

Syntax : set.clear()

```
>>> a={1,2}   → { }
>>> a.clear()
>>> print(a)
```

Dictionary

- Dictionary in Python is a data structure that stores data in the form of key-value pairs.
- Dictionary is unordered, mutable and indexed. It is used when we need to store data in an organized manner with key and value.
- Python में डिक्शनरी एक डाटा स्ट्रक्चर है, जो key-value pair के रूप में डेटा को स्टोर करता है।
- डिक्शनरी unordered, mutable और indexed होती है। इसका उपयोग तब किया जाता है जब हमें डेटा को कुंजी (key) और मान (value) के साथ संगठित रूप से संग्रहित करना हो।

The main features of the dictionary are:

1. Key-Value Pairs: Every key has a value.
2. Mutable: In this the data can be changed.
3. Keys are unique: A dictionary cannot have two identical keys.
4. Unordered: The order of the data is not fixed.

Syntax:

```
my_dict = {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
}
```

a = { "1" : "akash" , "2" : "vikas" }

Key value Key value

The main features of the dictionary are:

Key-Value Pairs: हर कुंजी का एक मान होता है।
2. Mutable: इसमें डेटा को बदला जा सकता है।
3. Keys Unique होती हैं: एक डिक्शनरी में दो समान कुंजी नहीं हो सकतीं।
4. Unordered: डेटा का क्रम तय नहीं होता।

Syntax:

```
my_dict = {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
}
```

Shallow Copy:

- Shallow copy only copies the reference of the original collection.
- If there is a nested object inside the collection (such as a list or a list inside a dictionary), then shallow copy copies only the reference in it, not the actual data.
- You can create a shallow copy using the copy() method or copy module.

Change

a → ( )

b → ( )

Change

- Shallow copy में सिर्फ original collection का reference कॉपी होता है।
- अगर collection के अंदर कोई nested object (जैसे list या dictionary के अंदर list) हो, तो shallow copy उसमें सिर्फ reference कॉपी करता है, actual data नहीं।
- Shallow copy को आप copy() method या copy module से बना सकते हैं।

Example:

[2[0],2[1]]

```
import copy

original_list = [1, 2, [3, 4]]
shallow_copy = copy.copy(original_list)

print("Original List:", original_list)        # Output: [1, 2, [3, 4]]
print("Shallow Copy:", shallow_copy)   # Output: [1, 2, [3, 4]]
```

0 1 2

Example:

```
import copy

original_list = [1, 2, [3, 4]]
shallow_copy = [1,2[3,4]]
shallow_copy[2][0] = 99

print("Original List after change:", original_list)
print("Shallow Copy after change:", shallow_copy)

# Output: [1, 2, [99, 4]]
# Output: [1, 2, [99, 4]]
```

2[0] 2[1]

[1 , 2 , [3 , 4]]

0 1 2

99

[1, 2, [99, 4]

Shallow Copy:

Deep Copy :

- In deep copy, the entire data is copied, including nested objects.
- In deep copy, the original collection and the copied collection are completely separate.
- copy.deepcopy() is used to create it.

- Deep copy में पूरा डेटा कॉपी किया जाता है, nested objects भी।
- Deep copy में original collection और copied collection पूरी तरह अलग होते हैं।
- इसे बनाने के लिए copy.deepcopy() का उपयोग किया जाता है।

```
>>> import copy
>>> a=[1, 2, 3, [4, 5]]
>>> DC= copy.deepcopy(a)
>>> DC[3][0] = 45;
>>> Print(a)
>>> [1, 2, 3, [4, 5]]
>>> Print(DC)
>>> [1, 2, 3 [45, 5]]
```

→ original list

→ DC only change its content

4-chapter ⇒ 25 marks

:- unit No-5 (function) :->

```
a = 20
b = 30
Print (a+b)

d = 10
e = 15
Print(d+e)

f = 5
g = 6
Print(f+g)
```

```
add (a,b)
  return a+b.

Print(add(20,30)

Print(add(10,15)

Print(add(5,6)
```

Function :

- A function is a block or group of code that can be used repeatedly. A function is used whenever we need to perform a particular task multiple times. Functions make the code organized and easy to understand.

- फ़ंक्शन (Function) एक ऐसा ब्लॉक या कोड का समूह होता है, जिसे बार-बार इस्तेमाल किया जा सकता है। जब भी हमें किसी विशेष कार्य को कई बार करना हो, तो फ़ंक्शन का उपयोग किया जाता है। फ़ंक्शन कोड को व्यवस्थित और समझने में आसान बनाते हैं।

Types of Function :

(1). Built-in Function :

Python has pre-built functions that you can use directly.
Examples:

print(): To print something on the screen.
len(): To know the length of a list, string, etc.
type(): To know the type of data.

a = 20
Print (type (a))
↓ integer

Python में पहले से बने हुए फ़ंक्शन्स होते हैं, जिन्हें आप सीधे उपयोग कर सकते हैं।
उदाहरण:

print(): स्क्रीन पर कुछ प्रिंट करने के लिए।
len(): किसी लिस्ट, स्ट्रिंग आदि की लंबाई जानने के लिए।
type(): डेटा का प्रकार जानने के लिए।

Types of Function :

(2). User definedFunction :

- A user-defined function in Python is a function created by the user to perform specific tasks. Python allows you to define your own functions using the def keyword.
- Python में यूज़र डिफ़ाइन फ़ंक्शन का मतलब ऐसा फ़ंक्शन है जिसे उपयोगकर्ता अपनी ज़रूरत के अनुसार बनाता है। इसे बनाने के लिए Python में def कीवर्ड का उपयोग किया जाता है।

Syntax:

```
def add(Parameter):
    Stmt 1;
    Stmt 2;
```

function_name:

- The name of the function (should be descriptive).
- फ़ंक्शन का नाम (यह आपके कार्य का वर्णन करना चाहिए)।

parameters:

- Input values passed to the function (optional).
- इनपुट वैल्यूज जिन्हें फ़ंक्शन में भेजा जाता है (optional)।

return:

- Returns the output (optional).
- फ़ंक्शन का आउटपुट (optional)।

Type

How to Define a Function:

(1). Function Without Parameters:

- Such functions are those in which no arguments are passed. These are also called parameter-less functions.
- ऐसे फंक्शन वह होते हैं जिनमें कोई भी argument पास नहीं किया जाता। इन्हें parameter-less functions भी कहा जाता है।

Keyword

JN_Ramp

Example:

```
def say_hello():
    print("Hello, welcome to Python!")

# Call the function
say_hello()
```

No Parameter

O/P = Hello, welcome to Python.

(2). Function With Parameters

- When we call a function, we can pass it some arguments (or parameters). These arguments are used inside the function.
- जब हम किसी फंक्शन को कॉल करते हैं, तो हम उसे कुछ arguments (या पैरामीटर्स) दे सकते हैं। इन arguments का उपयोग फंक्शन के अंदर किया जाता है।

a, b → argument

Example:

```
def add_numbers(a, b):
    return a + b

# Call the function
result = add_numbers(5, 10)
print("Sum:", result)
```

a b

→ 15

(3). Function With Default Parameters

- In Python, functions with default arguments are functions whose parameters are given a default value. If no value is given for that parameter while calling the function, the default value is used.
- Python में function with default argument वह फंक्शन होते हैं जिनके पैरामीटर को एक default value दी जाती है। यदि फंक्शन को कॉल करते समय उस पैरामीटर के लिए कोई वैल्यू नहीं दी जाती, तो डिफ़ॉल्ट वैल्यू का उपयोग किया जाता है।

```
def greet(a=3,b=4):
   c=a+b
   return c
# Call the function
print (default(6,7)
```

a = ~~3~~ 8
b = ~~4~~ 9 ⇒ 17

Print (greet(8,9))
17 //

(4). Function With Variable-Length Arguments

- To handle variable length arguments in Python you can use *args and **kwargs. Both of these methods allow you to dynamically handle the number of arguments inside a function.
- Python में variable length arguments को handle करने के लिए आप *args और **kwargs का उपयोग कर सकते हैं। ये दोनों तरीके आपको function के अंदर arguments की संख्या को dynamic रूप से handle करने की अनुमति देते हैं।

```
def multiply(*args):
    result = 1
    for num in args:
        result *= num
    return result
# Call the function
print(multiply(2, 3, 4))  # Output: 24
print(multiply(5, 10))    # Output: 50
```

(2, 3, 4)

1×2 = 2

2×3 = 6

6×4 = 24

(5). Function Returning Multiple Values

- To return multiple values from a function in Python, you can use a tuple, list, dictionary, or multiple return statements.

- Python में एक function से multiple values return करने के लिए आप tuple, list, dictionary, या multiple return statements का उपयोग कर सकते हैं।

```
def calculate(a, b):
    sum_val = a + b
    diff = a - b
    return sum_val, diff
```

[0] [1]

a = 10 = 15
b = 5
a-b = 10-5 = 5

```
# Call the function
result = calculate(10, 5)
print("Sum:", result[0])   # Output: Sum: 15
print("Difference:", result[1])  # Output: Difference: 5
```

Scope of Variable:

- Scope in Python means where and when a variable or function can be accessed.
- Python में scope का मतलब होता है कि किसी variable या function को कहाँ और कब एक्सेस किया जा सकता है।

(1). Local Scope:

- When you create a variable inside a function, that variable is available only inside that function. It cannot be accessed from outside.
- जब आप किसी function के अंदर एक variable बनाते हैं, तो वह variable उस function के अंदर ही उपलब्ध होता है। उसे बाहर से एक्सेस नहीं किया जा सकता।

```
def my_function():
    x = 10
    print(x)
my_function()
print(x)  # error
```

10

(2). Outer Enclosing Scope:

- This scope occurs when there is another function inside a function. The inner function can access the variables of the outer function, but the variables of the outer function are visible only in the outer function.
- यह scope तब होता है जब एक function के अंदर दूसरा function होता है। Inner function (अंदर वाला function) outer function के variables को एक्सेस कर सकता है, लेकिन outer function का variable सिर्फ outer function में ही visible होता है।

Example:

```
def outer_function():
    x = 20
    def inner_function():
        print(x)
    inner_function()

outer_function()
```

outer

inner

20

Global Scope :

- Global scope includes those variables which are created outside the function, i.e. anywhere in the entire program. You can access these variables anywhere in the entire code, whether you are inside or outside a function.
- Global scope में वो variables आते हैं जो function के बाहर, यानी पूरे program में कहीं भी बनाए गए होते हैं। आप इन variables को पूरे code में कहीं भी access कर सकते हैं, चाहे आप किसी function के अंदर हों या बाहर।

Example:

```
x = 50  # Global variable

def my_function():
    print(x)  # Global variable can be accessed inside function

my_function()  # Output: 50
print(x)  # Output: 50
```

First Class Citizens:

- In Python, a function is always a first class object. That is, we can pass a function as an argument to another function. And we can return a value in another function. For example, we can implement a switch case.

- Python में function हमेशा first class object होता है। अर्थात् हम एक function को दूसरे function के arguments के रूप में pass कर सकते हैं। तथा दूसरे function में value return कर सकते हैं। उदाहरण के तौर पर, हम switch case को implement कर सकते हैं।

```
def my ( )

def uncle (my)
```

Properties of first class functions in Python:

1. Function is a instance of object.
2. We can store function in a variables.
3. We can pass function as a parameter into another function.
4. We can return a function from function.

1. फ़ंक्शन ऑब्जेक्ट का एक उदाहरण है।
2. हम फ़ंक्शन को वैरिएबल में स्टोर कर सकते हैं।
3. हम फ़ंक्शन को दूसरे फ़ंक्शन में पैरामीटर के रूप में पास कर सकते हैं।
4. हम फ़ंक्शन से फ़ंक्शन वापस कर सकते हैं।

(1).Storing in a Variable:

- You can keep the function in a variable and call it later.
- function को एक variable में रख सकते हैं और बाद में उसे call कर सकते हैं।

Example:

```
def greet():
    print("Hello!")

my_func = greet
my_func()        # Output: Hello!
```

variable

(2).Passing a function as an argument:

- A function can be given as input to another function.
- function को दूसरे function में input के रूप में दे सकते हैं।

Example:

```
def greet():
    return "Hello!"

def print_message(func):
    print(func())

print_message(greet)  # Output: Hello!
```

(3).Returning the Function

- A function can return another function.
- एक function दूसरे function को return कर सकता है।

Example:

```
def outer():
    def inner():
        return "Hi!"
    return inner()

my_func = outer()
print(my_func())
```

Hi

imp

Map Function:

- map() is used when we need to perform the same type of operation on all the elements of a sequence.
- It takes a function and a sequence as input and applies the function to each item of the sequence.

- map() का उपयोग तब किया जाता है जब हमें किसी sequence के सभी elements पर एक ही प्रकार का operation करना हो।
- यह input में एक function और एक sequence लेता है और sequence के हर item पर function को apply करता है।

Syntax:

```
map(function, iterable)
```

a[1, 2, 3, 4]

a[2, 4, 6 8]

function:

- The function that will be applied to each element of the sequence.
- वो function जो sequence के हर element पर लागू होगा।

iterable:

- The sequence (such as list, tuple, etc.) on which the function will be applied.
- वो sequence (जैसे list, tuple, आदि) जिस पर function को लागू किया जाएगा।

Example 1:

```
def double(x):
    return x * 2
numbers = [1, 2, 3, 4]
result = map(double, numbers)
print(list(result))
```

1 x 2 = 2
2 x 2 = 4

[2, 4, 6, 8]

Example 2:

```
numbers = [1, 2, 3, 4]
result = map(lambda x: x * 2, numbers)
print(list(result))
```

Return

Parameter

[2, 4, 6, 8]

Filler:

- In Python, the filter() function is used to filter the elements of a sequence (such as a list, tuple, etc.). It returns only those elements that satisfy a given condition (logic).
- Python में filter() function का उपयोग एक sequence (जैसे list, tuple आदि) के elements को filter करने के लिए किया जाता है। यह केवल उन्हीं elements को return करता है जो किसी दिए गए condition (logic) को satisfy करते हैं।

Syntax:

filter(function, iterable)

⇒ a[1,2,3,4,5,6]

Print even → 2,4,6

function:

- A function that is applied to each element. It returns True or False.
- एक function जो हर element पर apply किया जाता है। यह True या False return करता है।

iterable:

- The sequence (e.g. list, tuple, set) that is to be filtered.
- वह sequence (जैसे list, tuple, set) जिसे filter करना है।

Example: Print the odd number list

```
def is_odd(num):
    return num % 2 != 0

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd_numbers = filter(is_odd, numbers)
print("Odd Numbers:", list(odd_numbers))
```

1 % 2 → 1
2 % 2 → 0
3 % 2 → 1
4 % 2 → 0
5 % 2 → 1

[1, 3, 5, 7, 9]

Example: Print the even number list

```
def is_even(num):
    return num % 2 == 0

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd_numbers = filte(is_even, numbers)
print("Odd Numbers:", list(odd_numbers))
```

1 . 1 . 2 => 1

2 . 2 == 0

2)2(1
2
0

[2, 4, 6, 8, 10]

Filter with lambda function:

- Lambda functions are used for small, temporary functions.
- Lambda functions छोटे, temporary functions के लिए उपयोग किए जाते हैं।

Example:

```
numbers = [10, 25, 30, 45, 50]
result = filter(lambda x: x > 30, numbers)
print(list(result))

# Output: [45, 50]
```

x > 30

10 > 30 → No
25 > 30 → No
30 > 30 → No
45 > 30 ✓ Yes
50 > 30 ✓

Example:

```
names = ["Alice", "Bob", "Charlie", "David"]
result = filter(lambda x: x.startswith("A"), names)
print(list(result))

# Output: ['Alice']
```

Mapping Functions in a Dictionary

- The mapping function in a dictionary in Python means establishing a relationship between a key and a value. A dictionary is a data structure that stores data in key-value pairs.
- Python में dictionary में mapping function का मतलब है key और value के बीच संबंध स्थापित करना। Dictionary एक data structure है जो key-value pairs में data को store करता है।

Example 1: Accessing the value:

```
student_marks = {
   "Rahul": 85,
   "Priya": 90,
   "Amit": 78
}
#Accessing the value:
print(student_marks["Rahul"]) # output: 85
```

Example 2: Updating Key-Value Mapping:

```
student_marks = {
    "Rahul": 85,
    "Priya": 90,
    "Amit": 78
}
#Updating Key-Value Mapping:
student_marks["Amit"] = 80
print(student_marks)
# Output: {'Rahul': 85, 'Priya': 90, 'Amit': 80, 'Sita': 88}
```

Example: Adding a new mapping:

```
student_marks = {
    "Rahul": 85,
    "Priya": 90,
    "Amit": 78
}
student_marks["Sita"] = 88
print(student_marks)

# Output: {'Rahul': 85, 'Priya': 90, 'Amit': 78, 'Sita': 88}
```

Lambda Function in Python:

- A lambda function in Python is an anonymous function that you can define in a single line.
- It is also called an inline function or single-expression function.
- Lambda functions are useful for small, temporary operations that do not need to be used frequently.

Syntax:

lambda arguments: expression

def my(argument):
return;

Lambda Function in Python:

- Python में lambda function एक anonymous (नामहीन) function है, जिसे आप एक ही लाइन में define कर सकते हैं।
- इसे inline function या single-expression function भी कहा जाता है।
- Lambda functions छोटे, temporary operations के लिए उपयोगी हैं, जिन्हें बार-बार उपयोग करने की आवश्यकता नहीं होती।

Syntax :

```
lambda arguments: expression
```

- **lambda:** This keyword defines the lambda function.
- यह keyword lambda function को define करता है।

- **arguments:** It can have 0 or more arguments.
- इसमें 0 या अधिक arguments हो सकते हैं।

- **expression:** It is a single expression which is evaluated and returns the result.
- यह एक single expression है, जिसे evaluate किया जाता है और उसका result return होता है।

Example: Addition of two number.

```
add = lambda a, b: a + b
result = add(5, 3)
print(result)
```

8

Example: Print only greater than 30 values

```
numbers = [10, 25, 30, 45, 50]
result = filter(lambda x: x > 30, numbers)
print(list(result))
# Output: [45, 50]
```

Example: Addition of two number.

```
add = lambda a, b: a + b
result = add(5, 3)
print(result)
```

Example: Print only greater than 30 values

```
numbers = [10, 25, 30, 45, 50]
result = filter(lambda x: x > 30, numbers)
print(list(result))
# Output: [45, 50]
```

Thank you!

Chapter No-1 }
Chapter No-5 }
Complete