AI/ML/Python

About the Author

- Created By: Mohammad Salman
- Experience: 19 Years +
- Designation: Corporate Trainer

Microsoft CERTIFIED Professional

Microsoft Specialist

Microsoft CERTIFIED Solutions Developer

Icons Used

Questions

Tools

Hands-on Exercise

Coding Standards

Questions?

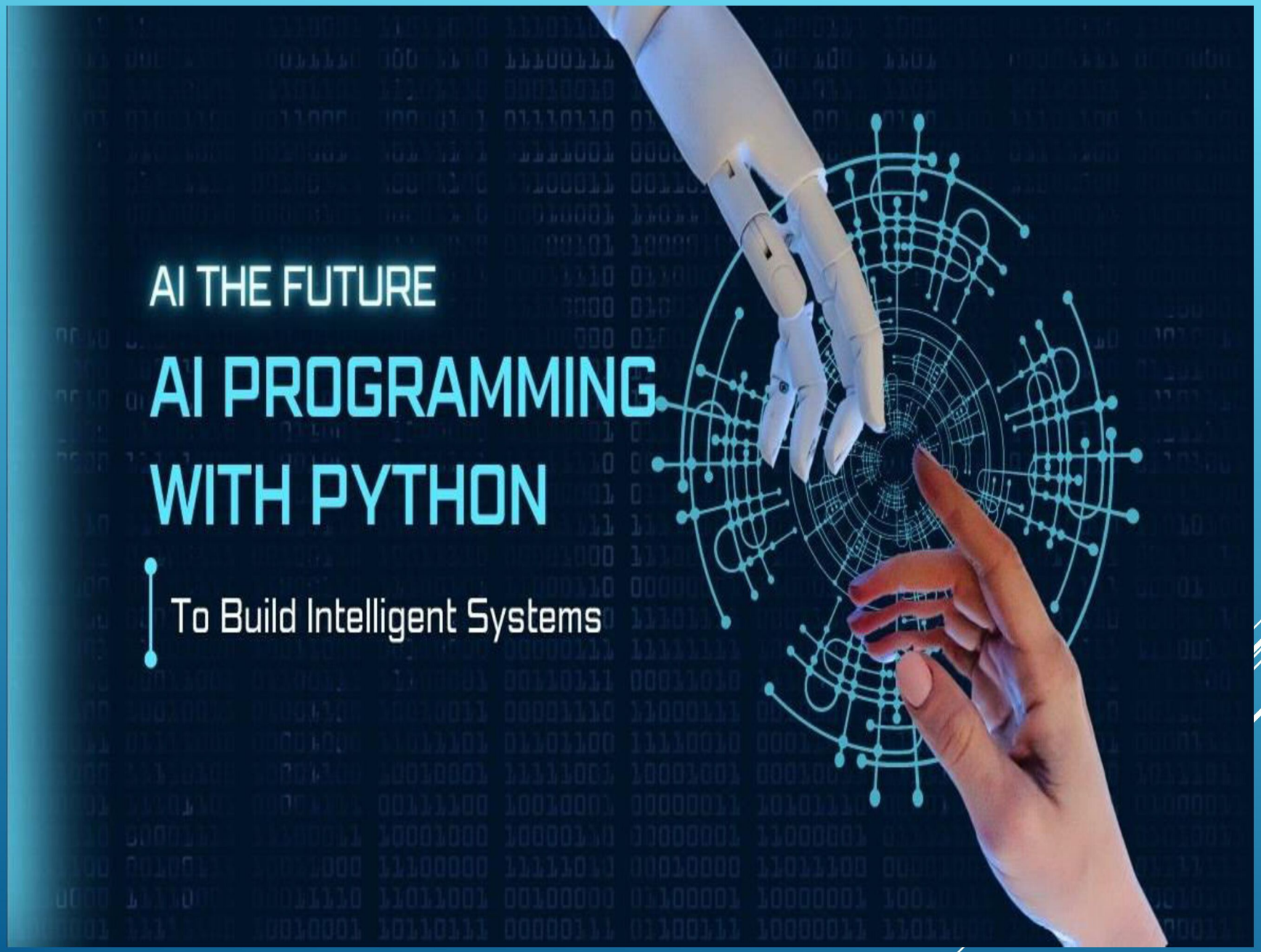
Reference

Try it Out

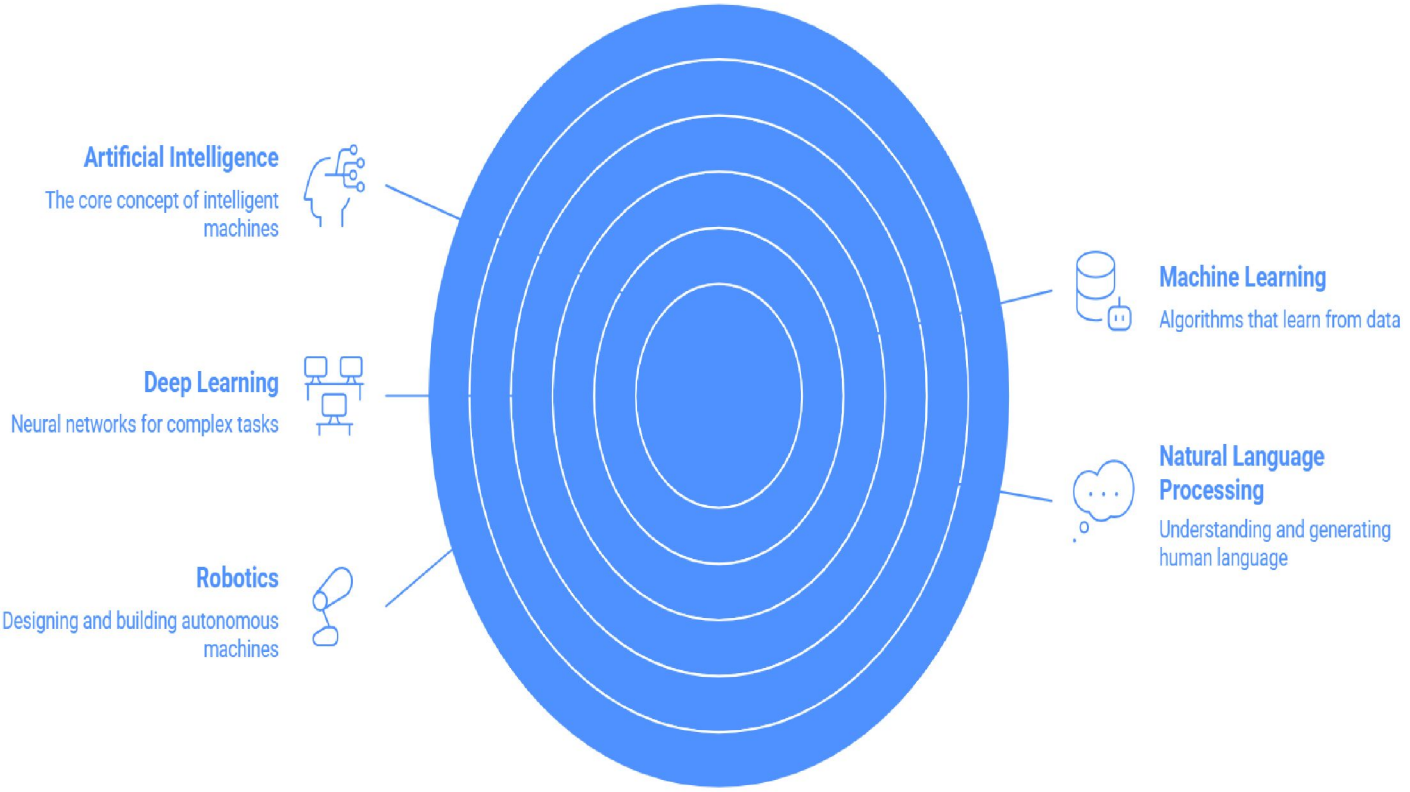
Informative Slide

Mandatory Slide

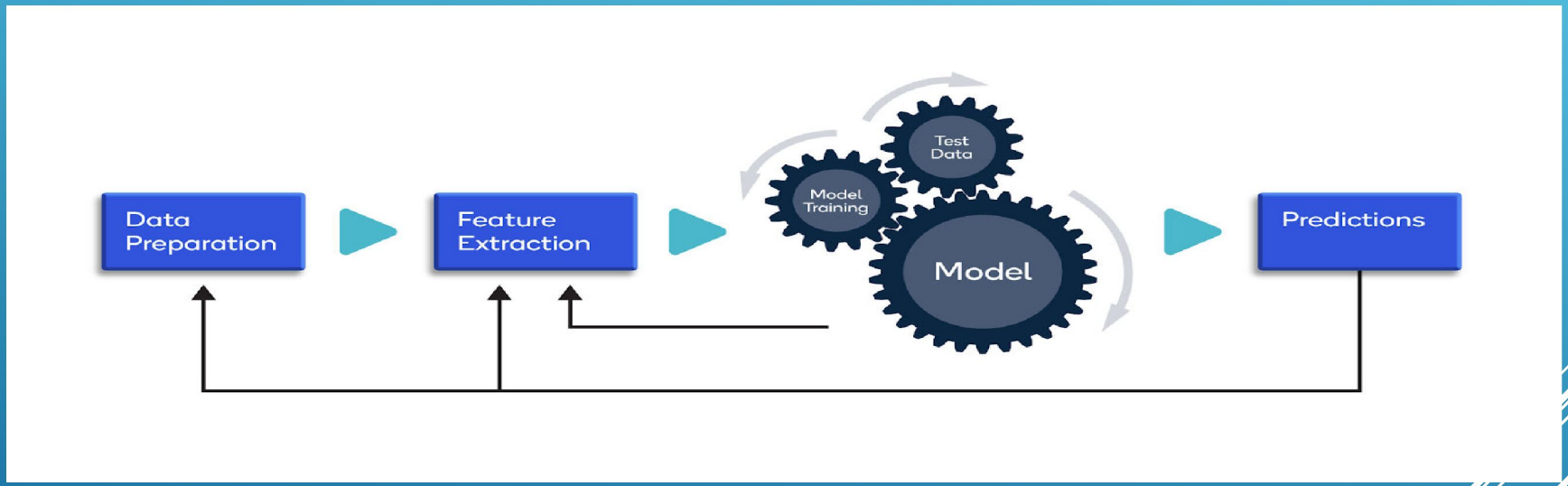
Welcome Break


AI THE FUTURE
AI PROGRAMMING
WITH PYTHON
To Build Intelligent Systems

AI and Its Subfields

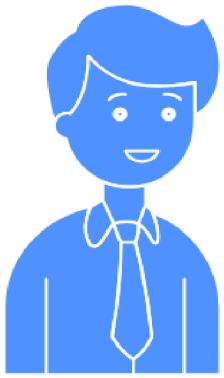


- Machine Learning (ML) is a subset of Artificial Intelligence that enables systems to learn from data and make predictions without explicit programming.



WHAT IS MACHINE LEARNING

Machine Learning Applications

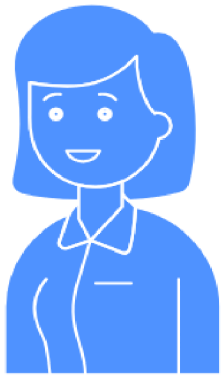


Where is machine learning used today?

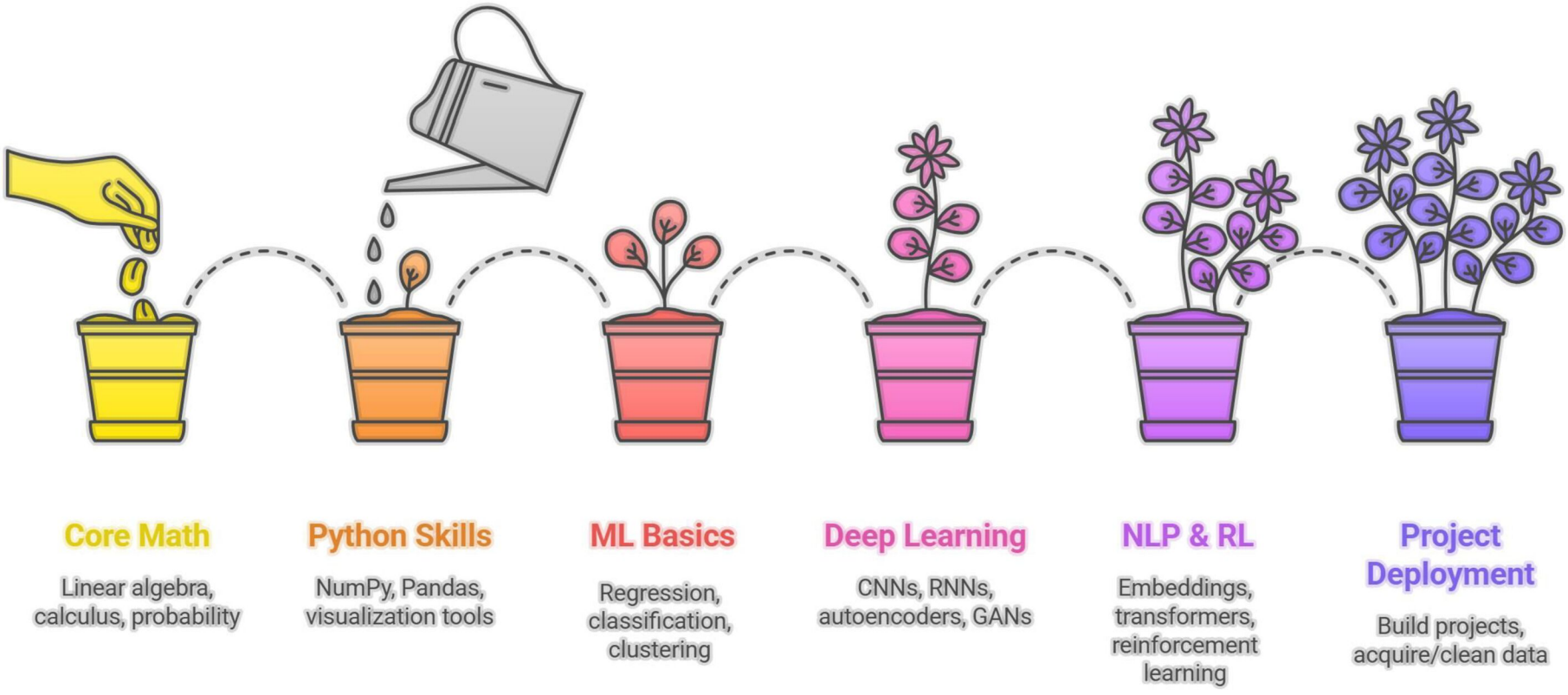
It's used in fraud detection, spam filters, product recommendations, medical diagnosis, and voice assistants.

What does machine learning do?

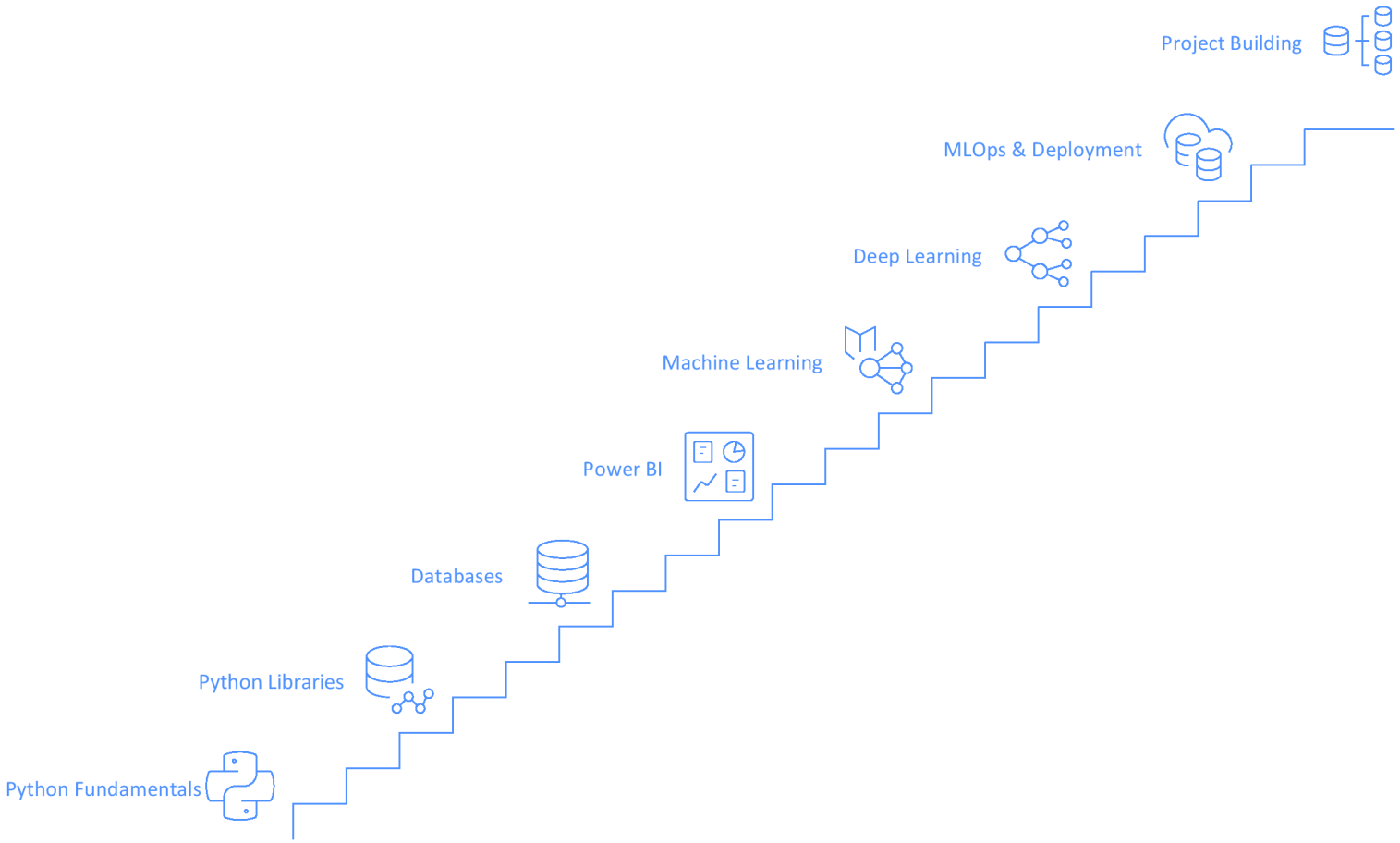
It automates decisions and learns from large datasets.



ROADMAP


Core Math
Linear algebra, calculus, probability
Python Skills
NumPy, Pandas, visualization tools
ML Basics
Regression, classification, clustering
Deep Learning
CNNs, RNNs, autoencoders, GANs
NLP & RL
Embeddings, transformers, reinforcement learning
Project Deployment
Build projects, acquire/clean data

Journey to AI Readiness



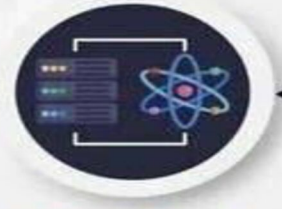
ML ENGINEER ROADMAP

Mathematics
Probability | Statictics | Discrete

Programming
Python | R | Java

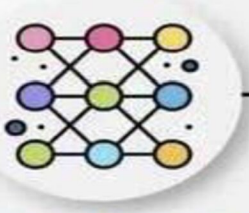
Database
MySQL | MongoDB

Machine Learning
Scikit learn Supervised learning Linsupervised learning | Reinforcement learning



Machine Learning
ML Libararies and Non-ML Libararies

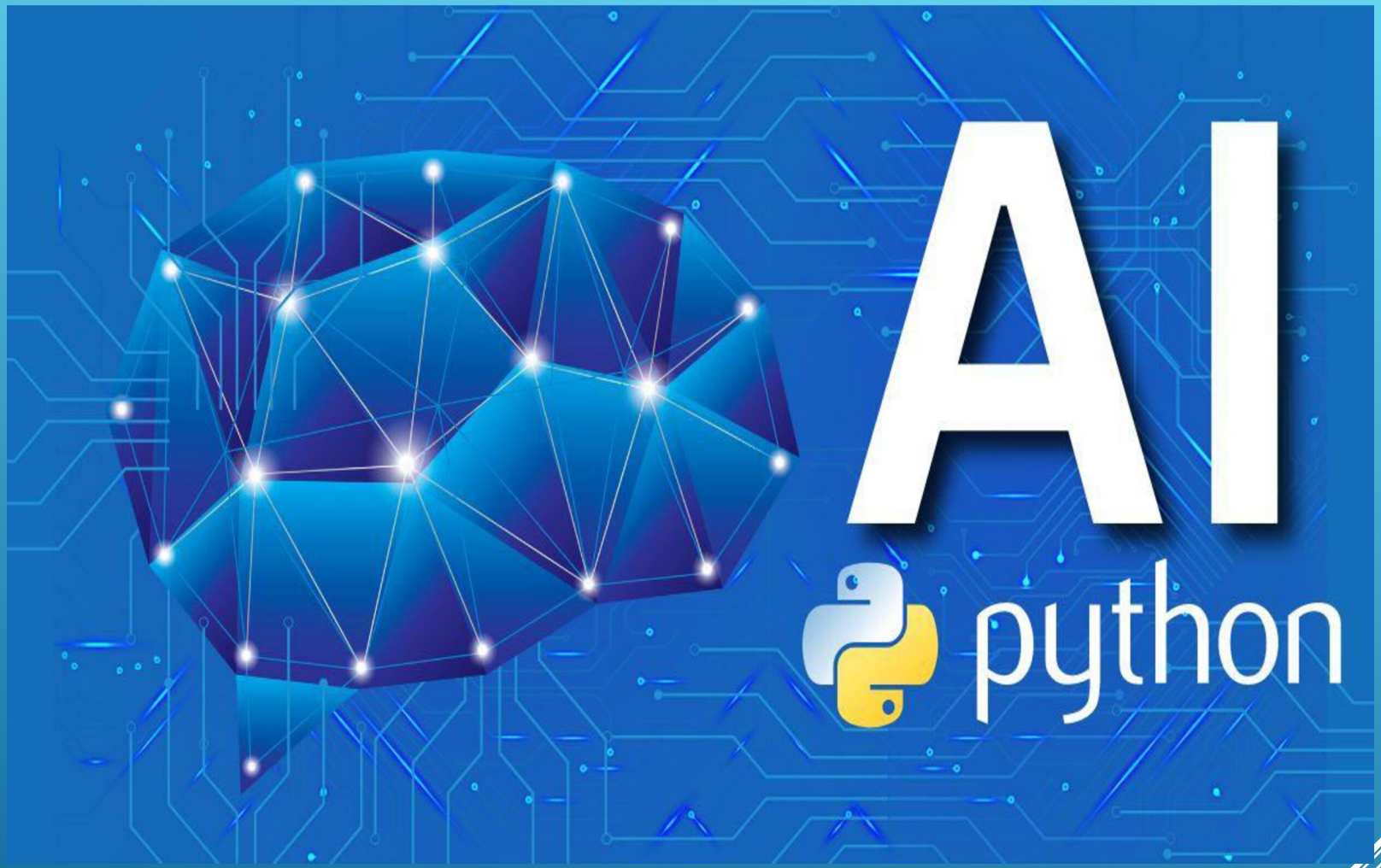
ML Algorithms
Linear | Logistic Regression | KNN | K-means | Random forest & more!



Deep Learning
TensorFlow, Keras | Neural Networks | CNN, RNN, GAN, LSTMS

Data Visualization Tools
Tableau | Qlikview | PowerBI

ML Engineer

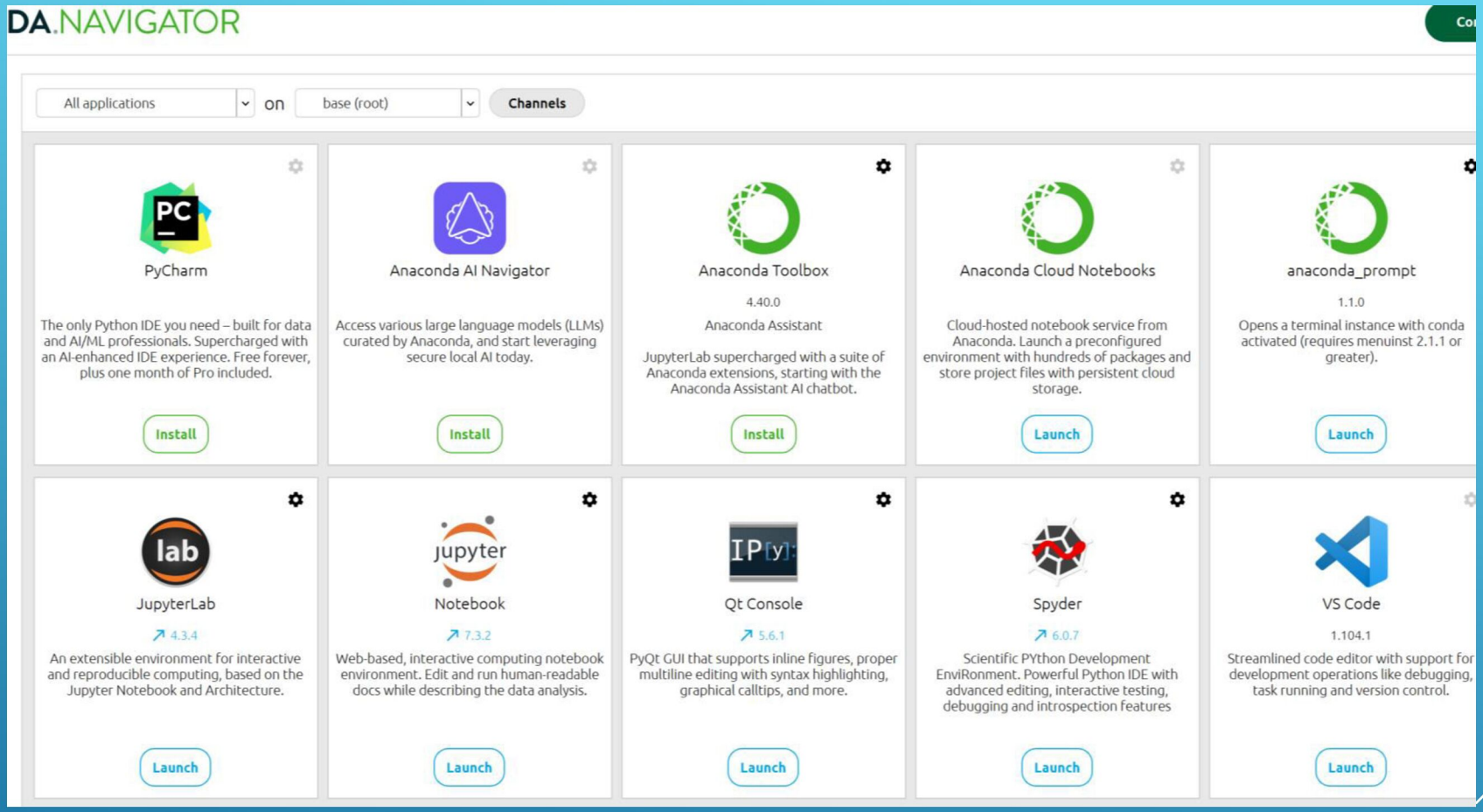


- Python is widely used in the field of artificial intelligence because of its simplicity and readability.
- Python is easy to learn.
- Python has a variety of libraries and frameworks, making it easier to develop AI applications.


Easy to Learn
Large Community
High-Level Language
Open Source
Multiple libraries
Interpreted Language
Versatile

- Why Python

LIBRARIES



MACHINE LEARNING SOFTWARE

TensorFlow

Google colab

ANACONDA

POWER BI

COPILOT

Install For Windows

1. Go to the official Python website https://www.python.org/downloads/
2. Download the latest versionYou'll see a big yellow button likeDownload Python 3.x.xClick it — it will automatically detect your Windows version (64-bit is most common).
3. Run the Installer (python-3.x.x.exe)When the installer opens: Check the box — "Add Python 3.x to PATH" (very important) Then click Install Now
4. Wait for installation to completeOnce it says "Setup was successful", close the window.
5. Verify installationOpen Command Prompt (CMD) or PowerShell, then type:

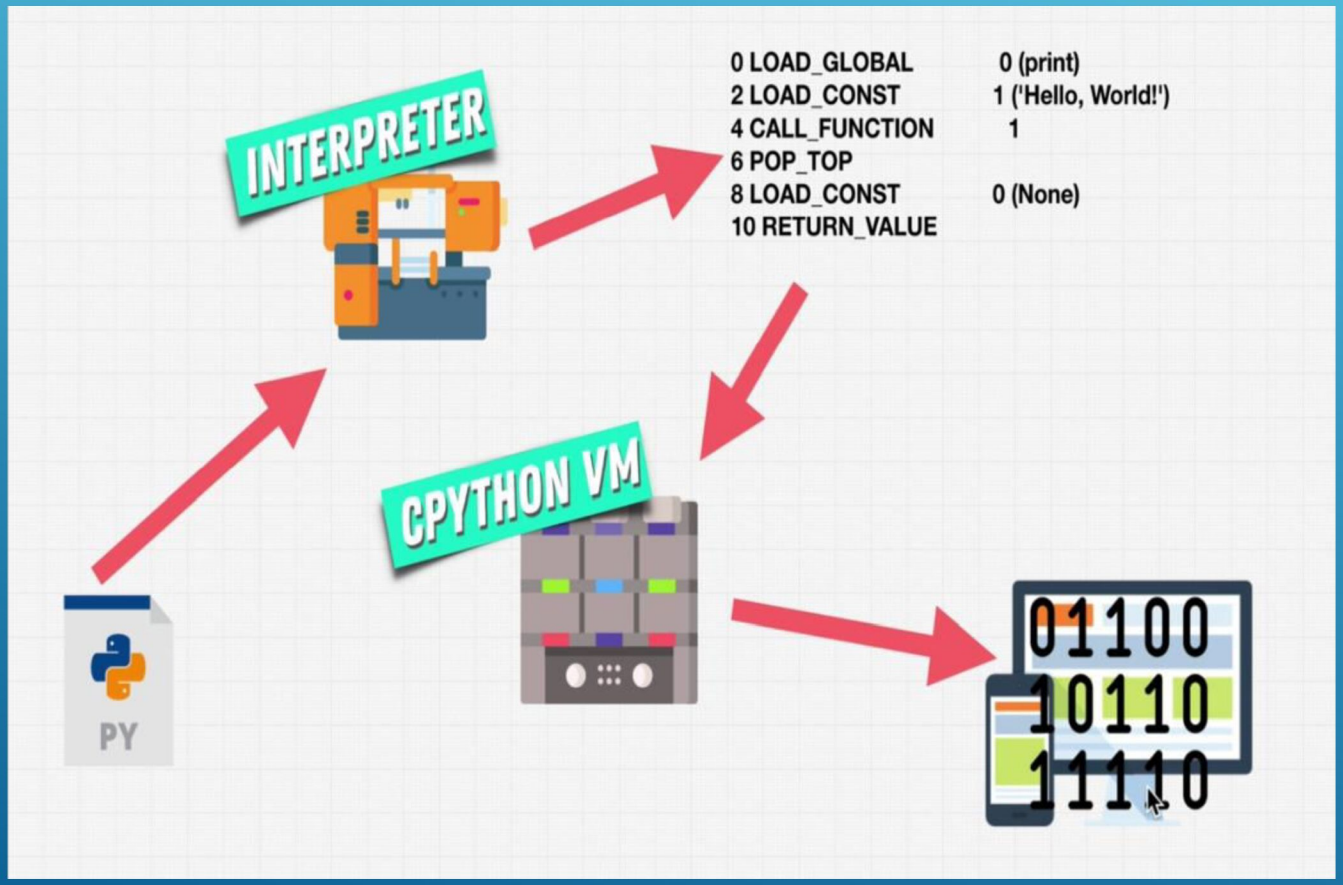
python --version

Install For macOS Users

1. Open the official Python site https://www.python.org/downloads/mac-osx/
2. Download the latest .pkg installerFor example: python-3.12.x-macos11.pkg
3. Open the downloaded fileFollow the installation steps → “Continue”, “Agree”, “Install”.
4. Verify installationOpen Terminal and type:

python3 --version

Python Interpreter

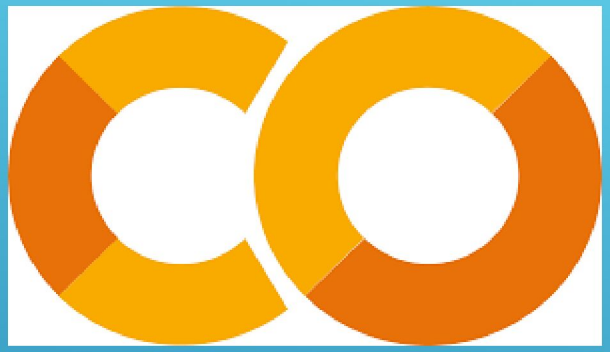


Google Collaboratory:

Or "Colab," is a free, hosted Jupiter notebook environment that requires no setup and runs entirely in the cloud.

It's particularly popular among data scientists, machine learning engineers, and students

Google Collaboratory:



Google Colab

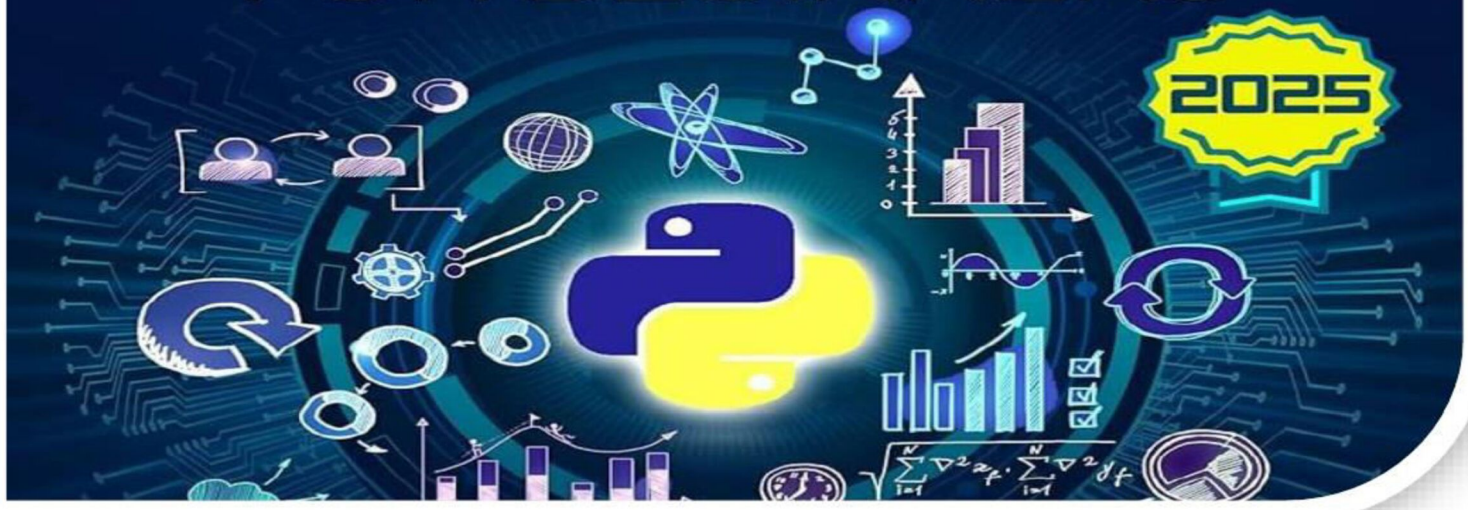
Pros

- Free access
- No setup
- Cloud-based
- Collaboration
- Pre-installed libraries

Cons

- Internet dependency
- Limited resources
- Privacy concerns

PYTHON PROGRAMMING FOR BEGINNERS



By: MOHD SALMAN

Python Syntax
Output:

```
print("Hello, World!")
print("Welcome to Python Programming")
print(10 + 5)
```

Python Syntax
Input:

```
name = input("Enter your name: ")
print("Hello,", name)
```

Python DataTypes

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview
None Type:	NoneType

Text Type

Represents textual data

Numeric Types

Includes integers, floats, and complex numbers

Sequence Types

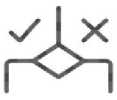
Consists of lists, tuples, and ranges

Mapping Type

Represents dictionaries

Set Types

Includes sets and frozensets



Boolean Type

Represents true or false values

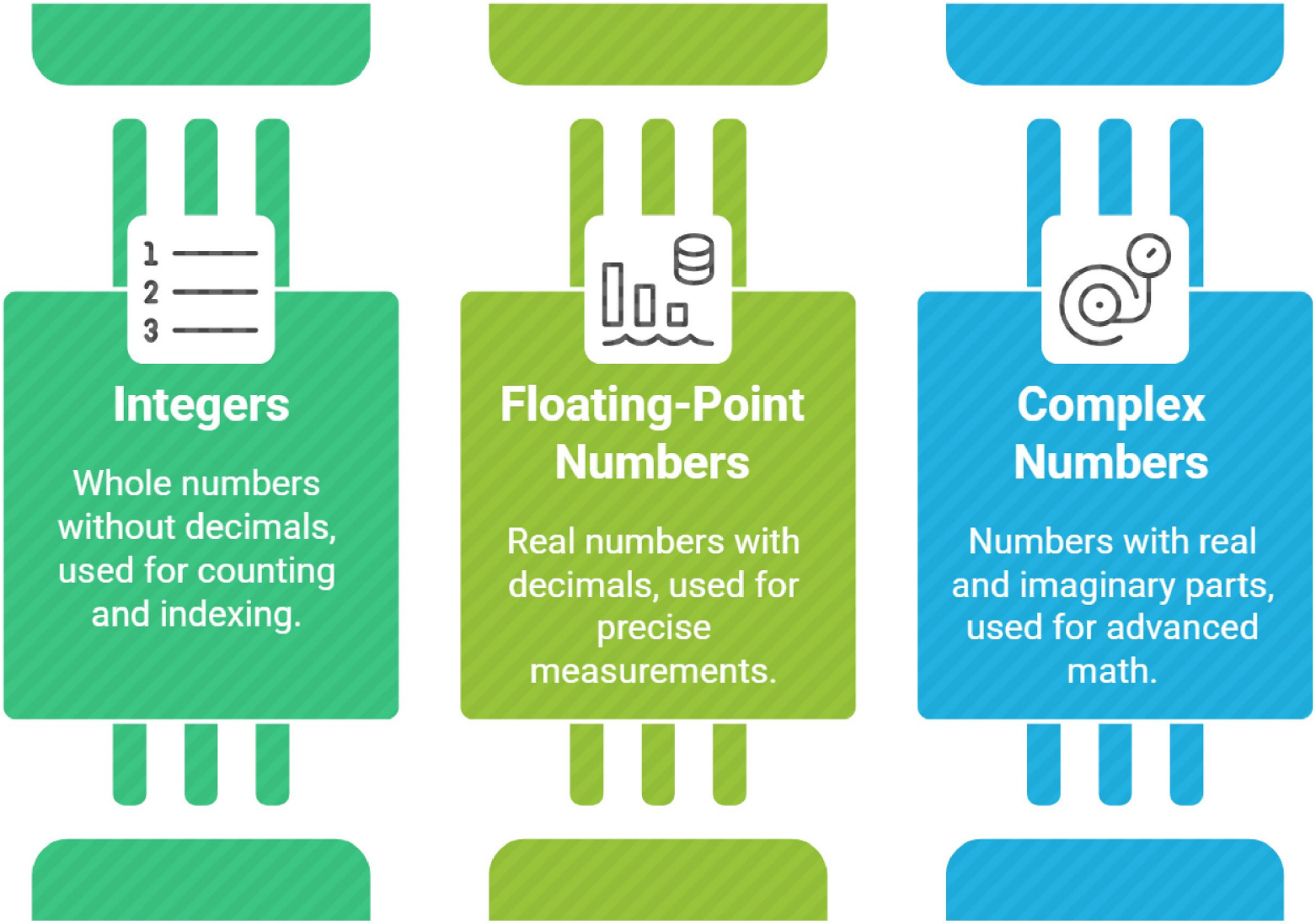
Binary Types

Consists of bytes, bytearrays, and memoryviews

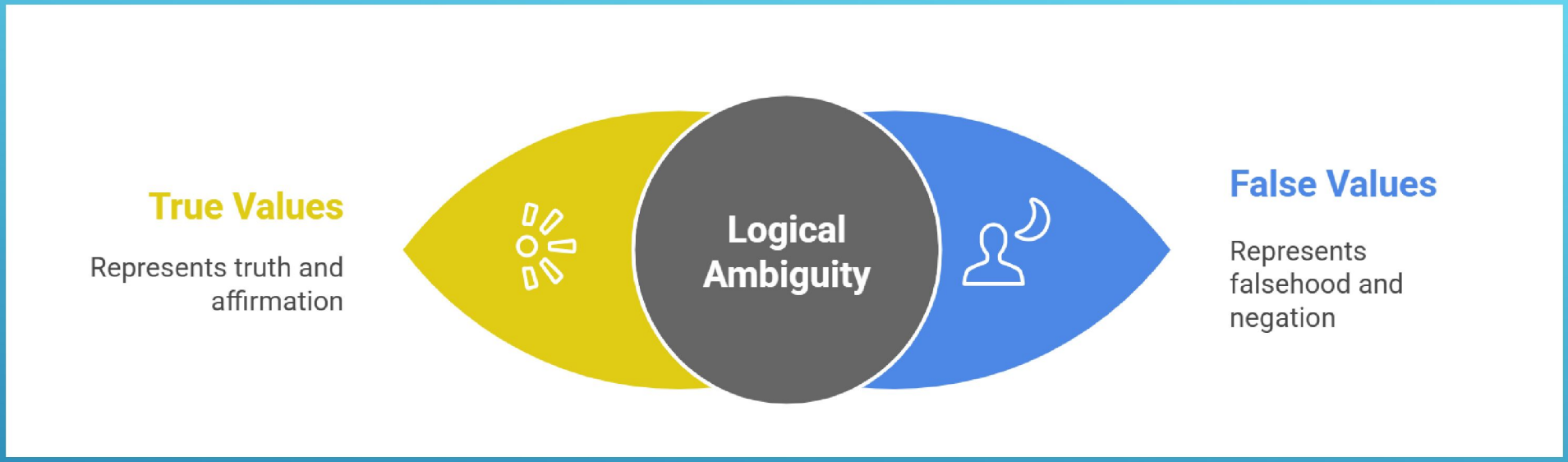
None Type

Represents the absence of a value

Python Numeric Types



Booleans (bool): True or False.




```
x = True
y = False
print(type(x))  # Output: <class 'bool'>
print(type(y))  # Output: <class 'bool'>
```

Logical Operations.

Logical AND

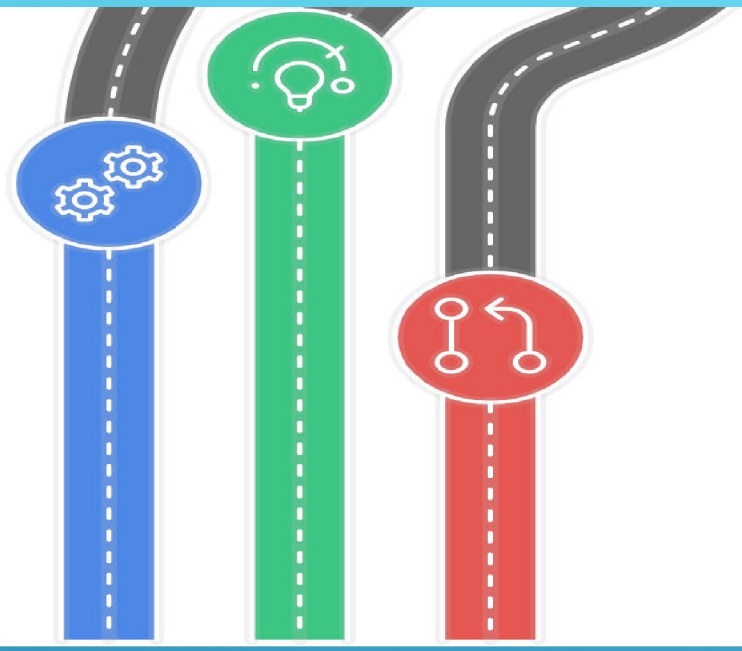
Returns True if both operands are True, otherwise False.

Logical OR

Returns True if at least one operand is True, otherwise False.

Logical NOT

Returns the opposite of the operand's value.



```
x = True
y = False
a = x and y  # Logical AND
b = x or y   # Logical OR
c = not x    # Logical NOT
print(a)  # Output: False
print(b)  # Output: True
print(c)  # Output: False
```

Python String Types

What are strings?

Strings represent sequences of characters.

Are strings mutable?

No, strings are immutable, meaning their values cannot be changed after creation.

Why are strings immutable?

This ensures that once a string is defined, it remains constant throughout its lifecycle, providing reliability in data handling and manipulation.

What happens when I try to modify a string?

Any operation that appears to modify a string will actually create a new string instance, preserving the original value intact.

String Manipulation Overview

Uppercase

Converts a string to uppercase.

Lowercase

Converts a string to lowercase.

Length

Determines the number of characters in a string.

Slicing

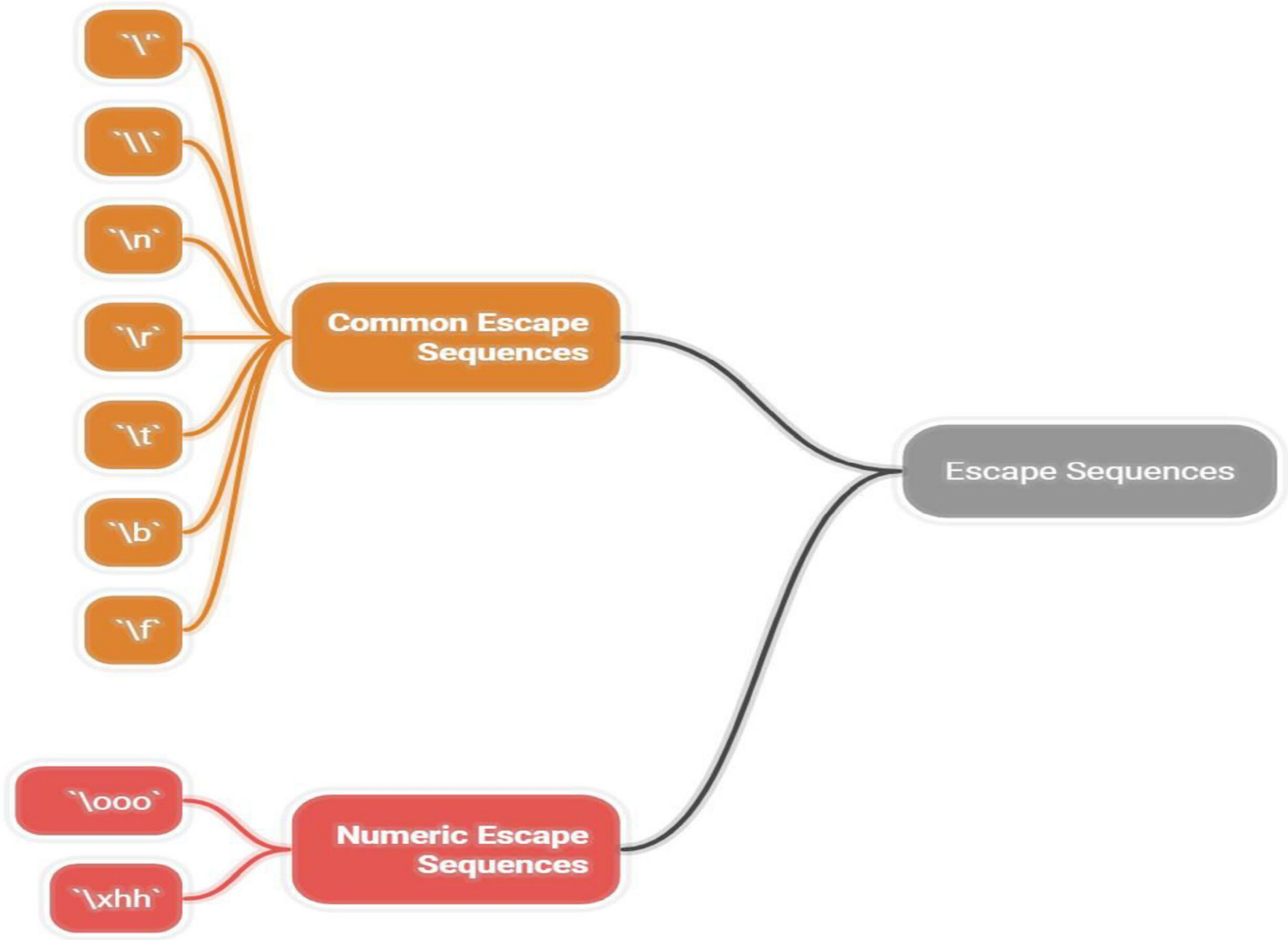
Extracts a substring from a string.

Concatenation

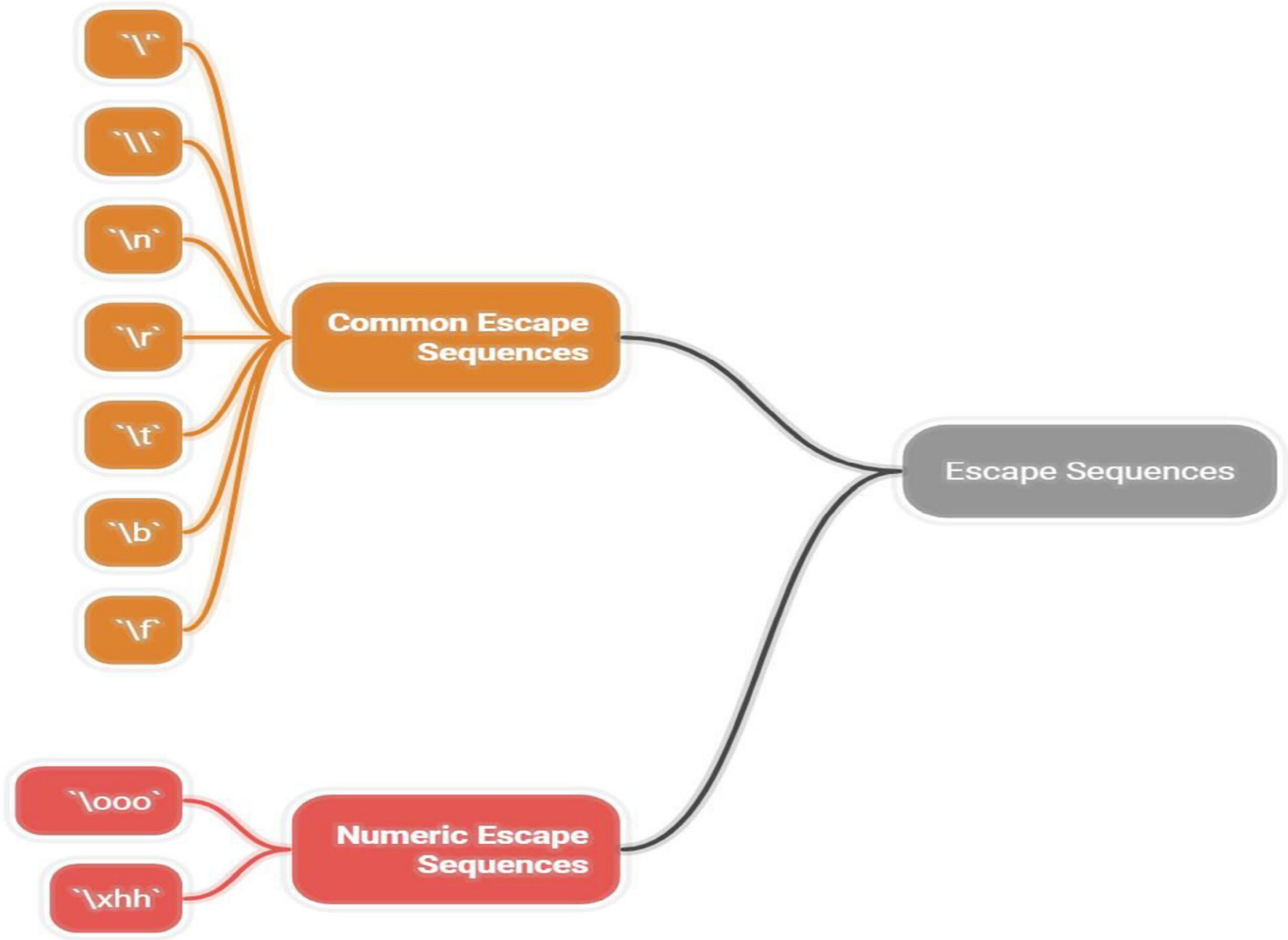
Combines two strings into one.

```
x="Hello"
y="World"
greeting = x + " " + y # Concatenation
print(greeting) # Output: Hello World
substring = x[0:2] # Slicing
print(substring) # Output: He
length = len(x) # Length
print(length) # Output: 5
uppercase = x.upper() # Uppercase
print(uppercase) # Output: HELLO
lowercase = x.lower() # Lowercase
print(lowercase) # Output: hello
```

String Escape Sequences



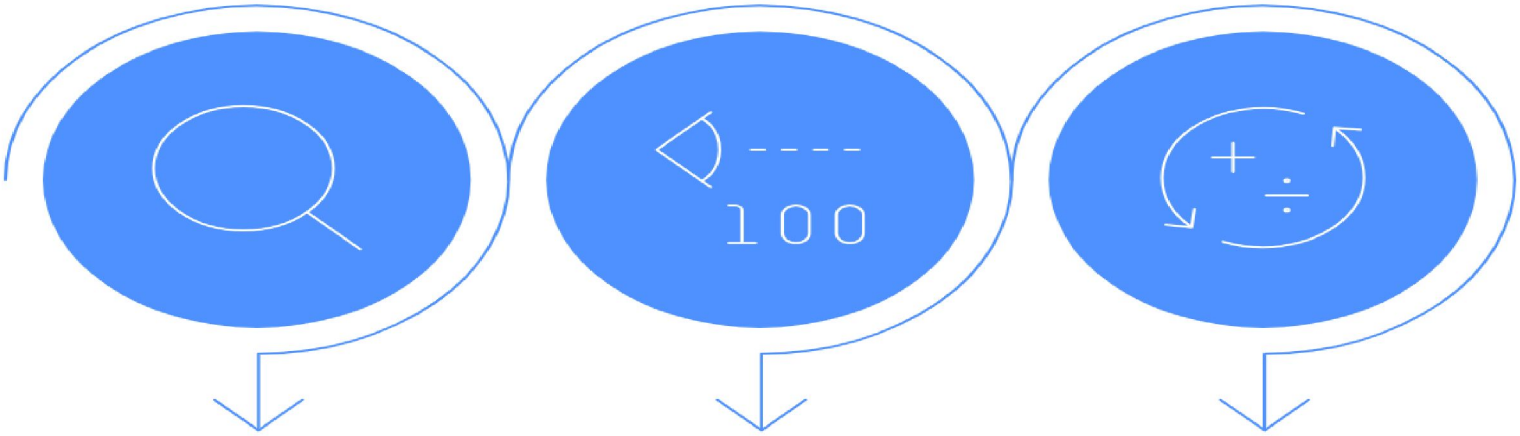
String Escape Sequences



Range

Ranges represent sequences of numbers.
They are often used in loops. It creates a range object, which you can loop through or convert into a list

```
my_range = range(5) # Generates numbers from 0 to 4
print(type(my_range)) # Output: <class 'range'>
my_list = list(my_range)
print(my_list) # Output: [0, 1, 2, 3, 4]
for i in range(3): print(i) # Output: 0, 1,
```



Range object

Generates a sequence of numbers. It can be used in loops or converted to a list.

Range example

The code generates numbers from 0 to 4. It then prints the type and converts it to a list.

Loop example

The code iterates through a range of 3. It prints each number in the range.

Collection Data Types

Collection Data Types

- List Example

```
colors = ["red", "green", "blue", "yellow"]
```

- Tuple Example

```
coordinates = (10.5, 20.3)
```

- Set Example

```
unique_numbers = {1, 2, 3, 4, 5}
```

- Dictionary Example

```
student = {    "name": "Alice",    "age": 21,    "grade": "A"}
```

Python Datatypes

Numeric Types

int :

int represents a whole number. It can be positive or negative and can be of any length.

int variables are created when you assign an integer value to them.

```
a = 10
#Must be whole Number, No Decimals

b = 123456789987654321
#Can be of any length

c = -721
#Can be +ve or -ve
```

int variables can also be created using int().

```
d = int(10)
#10 is assigned to d

e = int(12.21)
#only 12 is assigned to e, decimal value is ignored

f = int("123")
#123 is assigned to f

g = int("abc")
#Error : Invalid integer value. "abc" can't be converted to int
```

float :

float represents decimal point numbers. They can be positive or negative and can be of any length.

float variables are created when you assign decimal value to them.

```
a = 12.21
#Must be decimal point number

b = 123456789.987654321
#can be of any length

c = -45.67
#Can be +ve or -ve
```

float can also be used to represent scientific numbers with 'e' or 'E' indicating power of 10.

```
d = 854e5      #Valid
e = 17E21      #Valid
f = -65.23e6   #Valid
```

float variables can also be created using float() method.

```
g = float(12.21)
#12.21 is assigned to g

h = float("21.12")
#21.12 is assigned to h

i = float(10)
#10.0 is assigned to i
```

complex :

complex represents complex numbers with 'j' indicating imaginary part.

```
a = 5+7j
print(a)          #Output : (5+7j)
print(a.real)     #Output : 5.0
print(a.imag)     #Output : 7.0
```

Complex numbers can also constructed using complex() method.

```
a = complex(10)
print(a)          #Output : (10+0j)

b = complex(12, 21)
print(b)          #Output : (12+21j)
```

Text Types

str :

str represents strings in Python which are either enclosed within single quotation mark (' ') or double quotation mark (" ").

It is the only one type which represents texts in Python.

```
a = "String"   #Valid
b = 'String'   #Valid
```

Multi-line strings are represented by three double quotes or three single quotes.

```
a = """First Line
       Second Line
       Third Line
       ....
       ....
       And So On"""        #Valid

b = '''First Line
       Second Line
       Third Line
       ....
       ....
       And So On'''        #Valid
```

Like many other programming languages, strings in python are treated as an array of characters. So, you can retrieve individual characters or iterate whole string.

```
#Retrieving a character

a = "String"
print(a[2])        #Output : r

#Iterating using for loop

for x in "String" :
        print(x)
```

Python doesn't have a data type which represents a single character. Characters in Python are treated as str type with length 1.

Boolean Types

bool :

bool represents boolean types in Python which hold any one of two values : true or false.

```
a = True
b = False

c = 10 > 5
print(c)              #Output : True

d = 23.56
e = 78.12
print(d == e)         #Output : False

print("String" == "string")      #Output : False
```

None Type

None :

None keyword is used to represent a null value or no value.

```
a = None   #Variable 'a' contains no value

print(type(a))    #Output : <class 'NoneType'>
```

Collection Types

Collection types represents a collection of values.

list, tuple, set, dict are important collection types in Python.

All 4 have different usage and characteristics.

	list	tuple	set	dict
Indexed	Yes	Yes	No	No
Ordered	Yes	Yes	No	Yes
Changeable	Yes	No	No	Yes
Duplicates	Yes	Yes	No	No

list : Indexed, ordered, changeable and can have duplicates

```
stationeryList = ["Pen", "Pencil", "Eraser", "Pen", "Sharpner", "Pencil"]
```

tuple : Indexed, ordered, unchangeable and can have duplicates

```
nameTuple = ("John", "Arun", "Ben", "Zia", "Arun")
```

set : Unindexed, unordered, unchangeable and no duplicates

```
numberSet = {1.1, 2.2, 3.3, 4.4, 5.5}
```

dict : Unindexed, ordered, changeable and no duplicates

```
anyDictionary = {
                "One" : 1,
                2 : "Two",
                "Three" : 3.0,
                "Four" : True,
                }
```

Binary Types

Python provides 3 data types to handle binary data. They are bytes, bytearray and memoryview.

```
a = b"any_Char"
print(a)          #Output : b'any_Char'

b = bytearray(10)
print(b)
#Output :
bytearray(b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00')

c = memoryview(bytes(10))
print(c)
#Output : <memory at 0x0000000002E2B580>
```

type()

type() function is used to check the type of any variable.

```
a = 10
print(type(a))          #Output : <class 'int'>

b = 12.21
print(type(b))          #Output : <class 'float'>

c = complex(7, 5)
print(type(c))          #Output : <class 'complex'>

d = "String"
print(type(d))          #Output : <class 'str'>

e = 10 > 5
print(type(e))          #Output : <class 'bool'>

f = None
print(type(f))     #Output : <class 'NoneType'>
```