MPSSDEGB

कौशल से रोजगार, समृद्धि अपार

FITT

Foundation For Innovation And Technology Transfer

SANKALP

Ministry of Skill Development & Entrepreneurship

AI PROGRAMMING LANGUAGES AND TOOLS

OVERVIEW

Introduction to Python syntax

Features relevant to AI

Exploring Popular AI Libraries

- TensorFlow
- PyTorch

Basic AI algorithms in Python

Tools for different AI tasks

Introduction to Python syntax

PYTHON VARIABLES

- A variable is the name given to a memory location. A value-holding Python variable is also known as an identifier.
- Since Python is an infer language that is smart enough to determine the type of a variable, we do not need to specify its type in Python.
- Variable names must begin with a letter or an underscore, but they can be a group of both letters and digits.
- The name of the variable should be written in lowercase. Both Rahul and rahul are distinct variables.

Examples of valid identifiers: a123, _n, n_9, etc.
Examples of invalid identifiers: 1a, n%4, n 9, etc.

Introduction to Python syntax

PYTHON KEYWORDS

Python keywords are unique words reserved with defined meanings and functions that we can only apply for those functions.
Python's built-in methods and classes are not the same as the keywords. Built-in methods and classes are constantly present; however, they are not as limited in their application as keywords.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Introduction to Python syntax

PYTHON LITERALS

Python Literals can be defined as data that is given in a variable or constant.

String literals:

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes to create a string.

Numeric literals:

Numeric Literals are immutable. Numeric literals can belong to following four different numerical types.

Int(signed integers)	Long(long integers)	float(floating point)	Complex(complex)
Numbers(can be both positive and negative) with no fractional part. eg: 100	Integers of unlimited size followed by lowercase or uppercase L eg: 87032845L	Real numbers with both integer and fractional part eg: -26.2	In the form of a+bj where a forms the real part and b forms the imaginary part of the complex number. eg: 3.14j

Introduction to Python syntax

PYTHON OPERATORS

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Introduction to Python syntax

PYTHON COMMENTS

- Single-Line Comments

  ```
  # This code is to show an example of a single-line comment
  ```

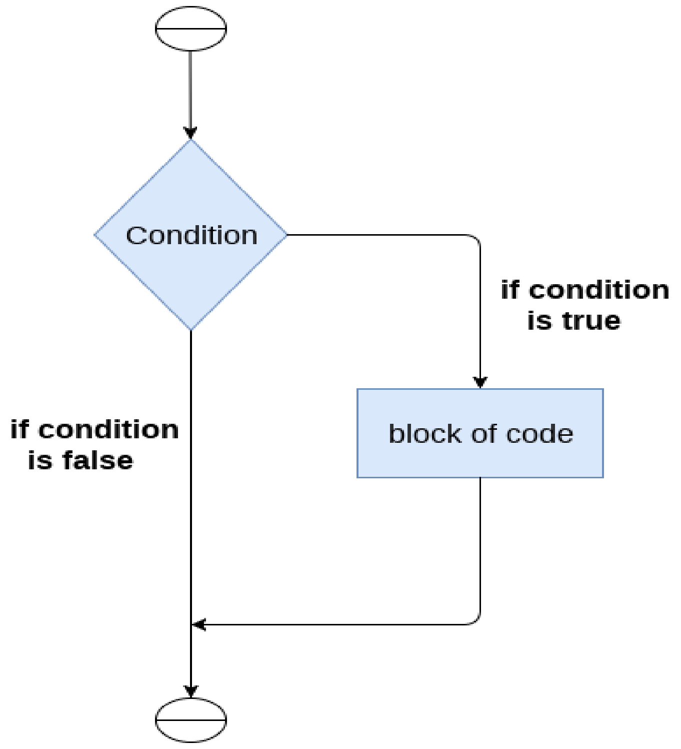
- Multi-Line Comments

  ```
  # it is a
  # comment
  # extending to multiple lines
  ```

Introduction to Python syntax

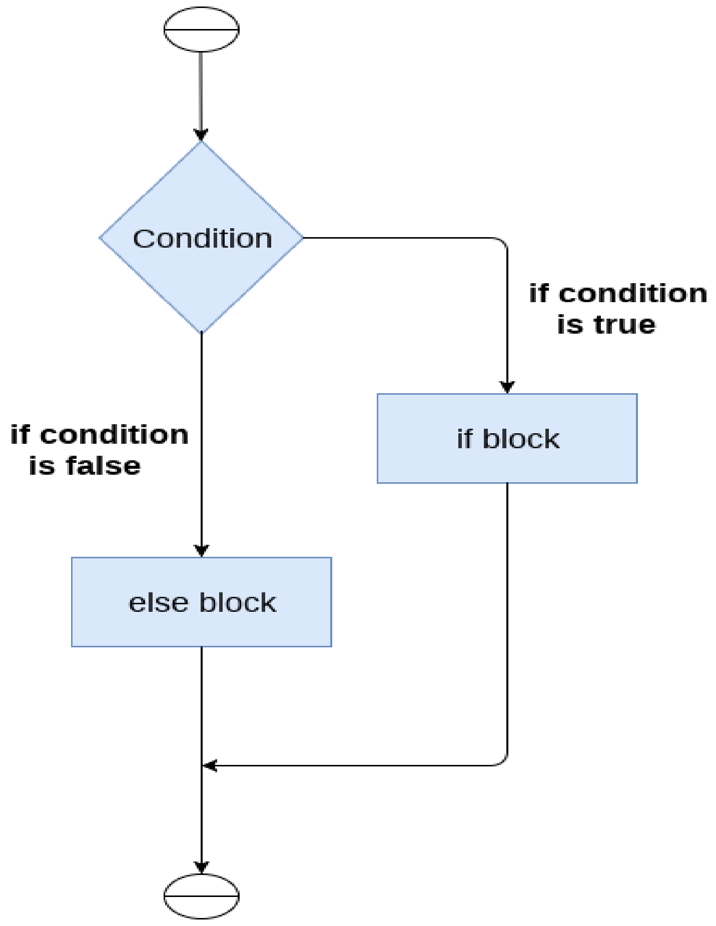
IF-ELSE

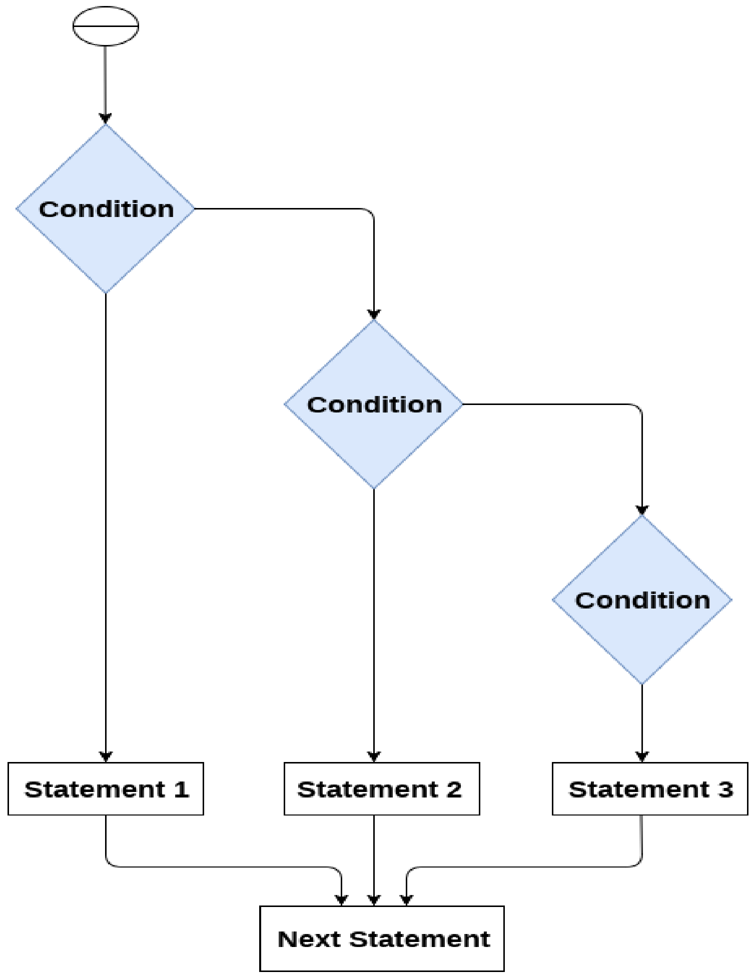
Statement	Description
If Statement	The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed.
If - else Statement	The if-else statement is similar to if statement except the fact that, it also provides the block of the code for the false case of the condition to be checked. If the condition provided in the if statement is false, then the else statement will be executed.
Nested if Statement	Nested if statements enable us to use if ? else statement inside an outer if statement.



Introduction to Python syntax

IF-ELSE





Introduction to Python syntax

PYTHON LOOPS

Sr.No.	Name of the loop	Loop Type & Description
1	**while loop**	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	**for loop**	This type of loop executes a code block multiple times and abbreviates the code that manages the loop variable.
3	**Nested loops**	We can iterate a loop inside another loop.

Introduction to Python syntax

PYTHON LOOPS

```
for value in sequence:
    { code block }
```

Python program to show how the for loop works

```
# Creating a sequence which is a tuple of numbers
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
# variable to store the square of the number
square = 0
# Creating an empty list
squares = []
 # Creating a for loop
for value in numbers:
    square = value ** 2
    squares.append(square)
print("The list of squares is", squares)
```

PYTHON LOOPS

```
for value in sequence:
     # executes the statements until sequences are exhausted
else:
     # executes these statements when for loop is completed

# Python program to show how to use else statement with for loop

# Creating a sequence
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)

# Initiating the loop
for value in tuple_:
    if value % 2 != 0:
        print(value)
# giving an else statement
else:
    print("These are the odd numbers present in the tuple")
```

PYTHON LOOPS

```
while <condition>:
    { code block }
```

```
# Python program to show how to use a while loop
counter = 0
# Initiating the loop
while counter < 10: # giving the condition
    counter = counter + 3
    print("Python Loops")
```

```
# Python program to show how to write a single state
ment while loop
counter = 0
while (count < 3): print("Python Loops")
```

```
#Python program to show how to use else statement with the while loop
counter = 0
# Iterating through the while loop
while (counter < 10):
    counter = counter + 3
    print("Python Loops") # Executed untile condition is met
# Once the condition of while loop gives False this statement will be execut
ed
else:
    print("Code block inside the else statement")
```

Introduction to Python syntax

BREAK, CONTINUE, PASS

Sr.No.	Name of the control statement	Description
1	**Break statement**	This command terminates the loop's execution and transfers the program's control to the statement next to the loop.
2	**Continue statement**	This command skips the current iteration of the loop. The statements following the continue statement are not executed once the Python interpreter reaches the continue statement.
3	**Pass statement**	The pass statement is used when a statement is syntactically necessary, but no code is to be executed.

BREAK, CONTINUE, PASS

Continue

```
# Initiating the loop
for string in "Python Loops":
    if string == "o" or string == "p" or string == "t":
        continue
    print('Current Letter:', string)
```

break

```
# Initiating the loop
for string in "Python Loops":
    if string == 'L':
        break
    print('Current Letter: ', string)
```

pass

```
for a string in "Python Loops":
  pass
print( 'Last Letter:', string)
```

Introduction to Python syntax

PYTHON STRINGS

- Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.
- The computer does not understand the characters; internally, it stores manipulated character as the combination of the 0's and 1's.
- Each character is encoded in the ASCII or Unicode character. So we can say that Python strings are also called the collection of Unicode characters.
- In Python, strings can be created by enclosing the character or the sequence of characters in the quotes.
- Python allows us to use single quotes, double quotes, or triple quotes to create the string.

Introduction to Python syntax

PYTHON LISTS

- In Python, the sequence of various data types is stored in a list. A list is a collection of different kinds of values or items.
- Since Python lists are mutable, we can change their elements after forming.
- The comma (,) and the square brackets [enclose the List's items] serve as separators.
- A list, a type of sequence data, is used to store the collection of data. Tuples and Strings are two similar data formats for sequences.
- Lists written in Python are identical to dynamically scaled arrays defined in other languages, such as Array List in Java and Vector in C++.

```
# a simple list
list1 = [1, 2, "Python", "Program", 15.9]
list2 = ["Amy", "Ryan", "Henry", "Emma"]
# printing the list
print(list1)
print(list2)
# printing the type of list
print(type(list1))
print(type(list2))
```

Introduction to Python syntax

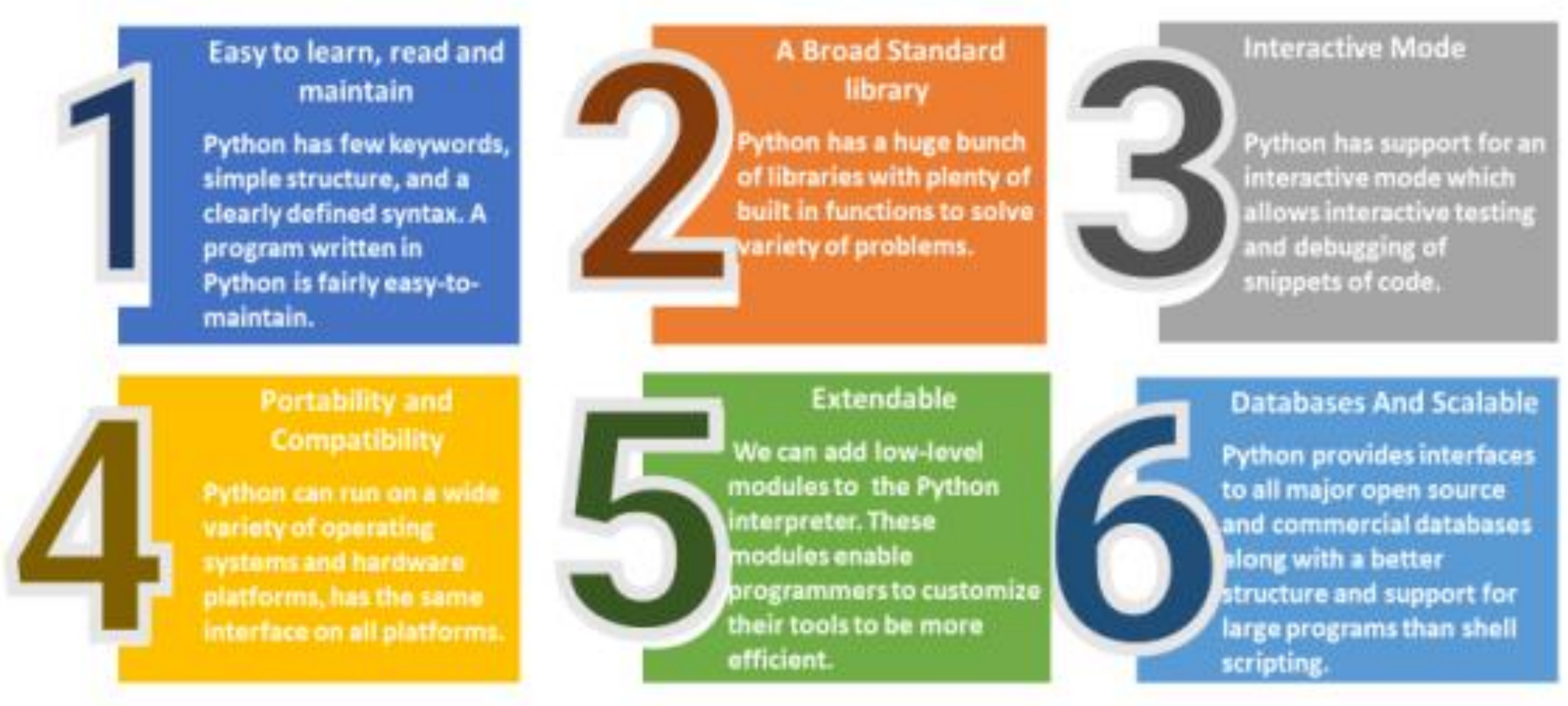
PYTHON TUPLES

- A comma-separated group of items is called a Python tuple.
- The ordering, settled items, and reiterations of a tuple are to some degree like those of a rundown, but in contrast to a rundown, a tuple is unchanging.
- The main difference between the two is that we cannot alter the components of a tuple once they have been assigned.
- On the other hand, we can edit the contents of a list.

```
# Python program to show how to create a tuple
# Creating an empty tuple
empty_tuple = ()
print("Empty tuple: ", empty_tuple)
# Creating tuple having integers
int_tuple = (4, 6, 8, 10, 12, 14)
print("Tuple with integers: ", int_tuple)
 # Creating a tuple having objects of different data typ
es
mixed_tuple = (4, "Python", 9.3)
print("Tuple with different data types: ", mixed_tuple)
```

Artificial Intelligence

- Artificial intelligence is the trending technology of the future
- There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI.
- Out of these, Python gains a maximum popularity because of the following reasons:



1 **Easy to learn, read and maintain**

Python has few keywords, simple structure, and a clearly defined syntax. A program written in Python is fairly easy-to-maintain.

2 **A Broad Standard library**

Python has a huge bunch of libraries with plenty of built in functions to solve variety of problems.

3 **Interactive Mode**

Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

4 **Portability and Compatibility**

Python can run on a wide variety of operating systems and hardware platforms, has the same interface on all platforms.

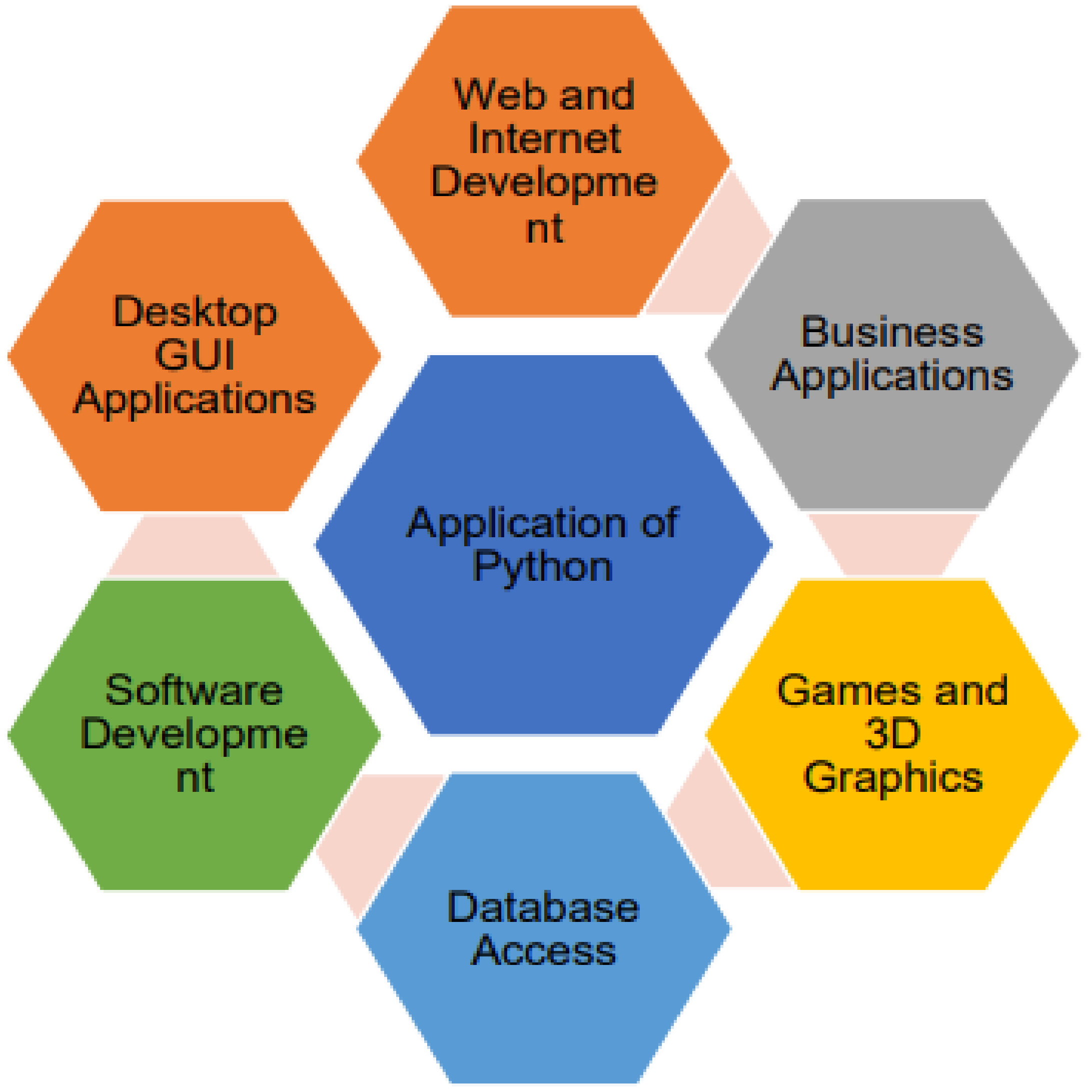
5 **Extendable**

We can add low-level modules to the Python interpreter. These modules enable programmers to customize their tools to be more efficient.

6 **Databases And Scalable**

Python provides interfaces to all major open source and commercial databases along with a better structure and support for large programs than shell scripting.

Applications of Python:



Artificial Intelligence and Python:

- Python comes with Prebuilt Libraries such as Numpy to perform scientific calculations, Scipy for advanced computing, and Pybrain for machine learning (Python Machine Learning), making it among the top languages for AI.

- Python developers all over the globe offer extensive support and assistance through tutorials and forums, helping the programmer much easier than another popular language.

- Python is platform-independent and therefore is among the most adaptable and well-known options for various platforms and technologies, with minimal modifications to the basics of coding.

- Python has the greatest flexibility among other programs, with the option of choosing among OOPs method and scripting. Additionally, you can use the IDE to search for all codes and be a blessing to developers struggling with different algorithms.

NumPy is used to store generalized data, which consists of one N-dimensional array and tools to integrate C/C++ codes, Fourier transformation, random numbers capabilities, and many other functions.

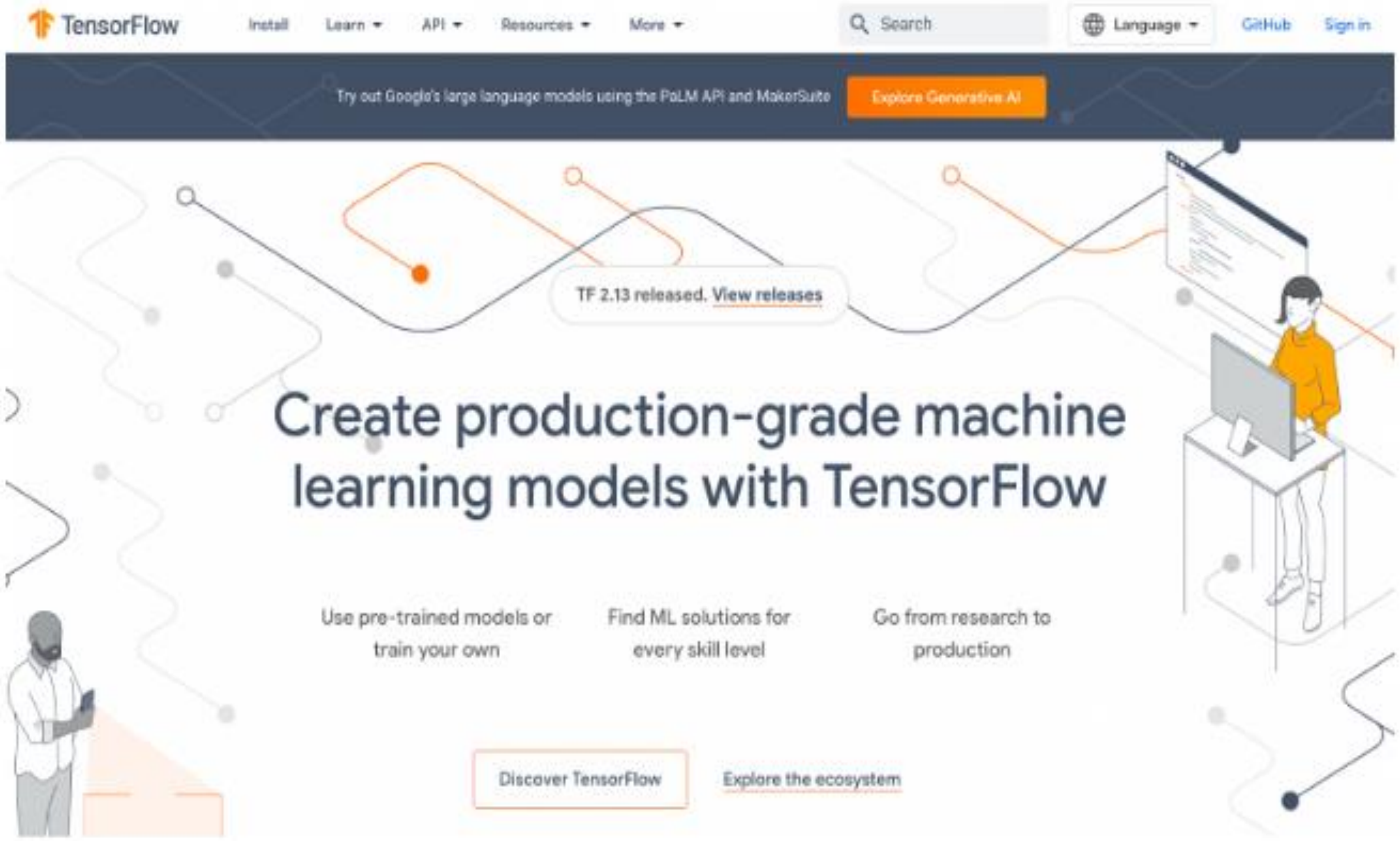
Another library to look into is pandas, an open-source library that provides users with data structures that are simple to use and analytical instruments that work with Python.
Matplotlib is a different service that is a plotting library for 2D producing high-quality publications. Matplotlib can be used to access up to six graphical users interface tools, Web application servers as well as Python scripts.

The most frequently used Python AI libraries are AIMA, pyDatalog, SimpleAI, EasyAi, and others. There are numerous Python machine learning libraries, such as PyBrain, MDP, scikit, PyML.

Features of AI

- A great library ecosystem
- A low entry barrier
- Flexibility
- Platform independence
- Readability
- Good visualization options
- Community support
- Growing popularity

TENSORFLOW



TensorFlow Install Learn API Resources More Search Language GitHub Sign in

Try out Google's large language models using the PaLM API and MakerSuite Explore Generative AI

TF 2.13 released. View releases

Create production-grade machine learning models with TensorFlow

Use pre-trained models or train your own

Find ML solutions for every skill level

Go from research to production

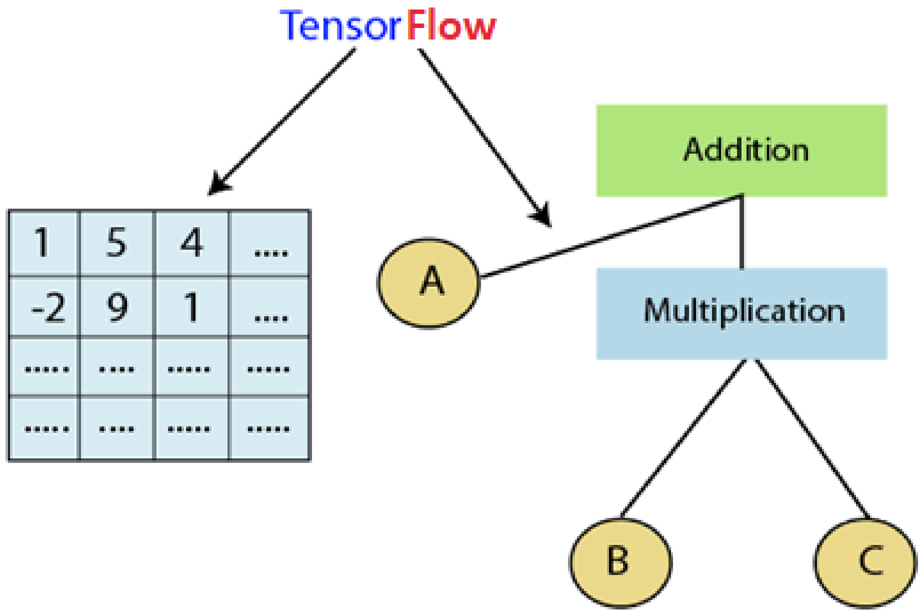
Discover TensorFlow

Explore the ecosystem

The word TensorFlow is made by two words, i.e., Tensor and Flow

- Tensor is a multidimensional array
- Flow is used to define the flow of data in operation.

TensorFlow is used to define the flow of data in operation on a multidimensional array or Tensor.



TensorFlow is a free, open-source Python library developed by Google.

It was first released in 2015, while the first stable version was coming in **2017**.

We can use it, modify it, and reorganize the revised version for free without paying anything to Google.

This is due to its ability to perform numerical computations at a high level.

Its flexible architecture and framework make it a popular choice in the machine-learning community.

It has other features like diverse platform support, immediate iterations, and data pipeline creation.

Features of TensorFlow

01. Responsive Construct

02. Flexible

03. Easily Trainable

04. Parallel Neural Network Training

05. Open Source

06. Large Community

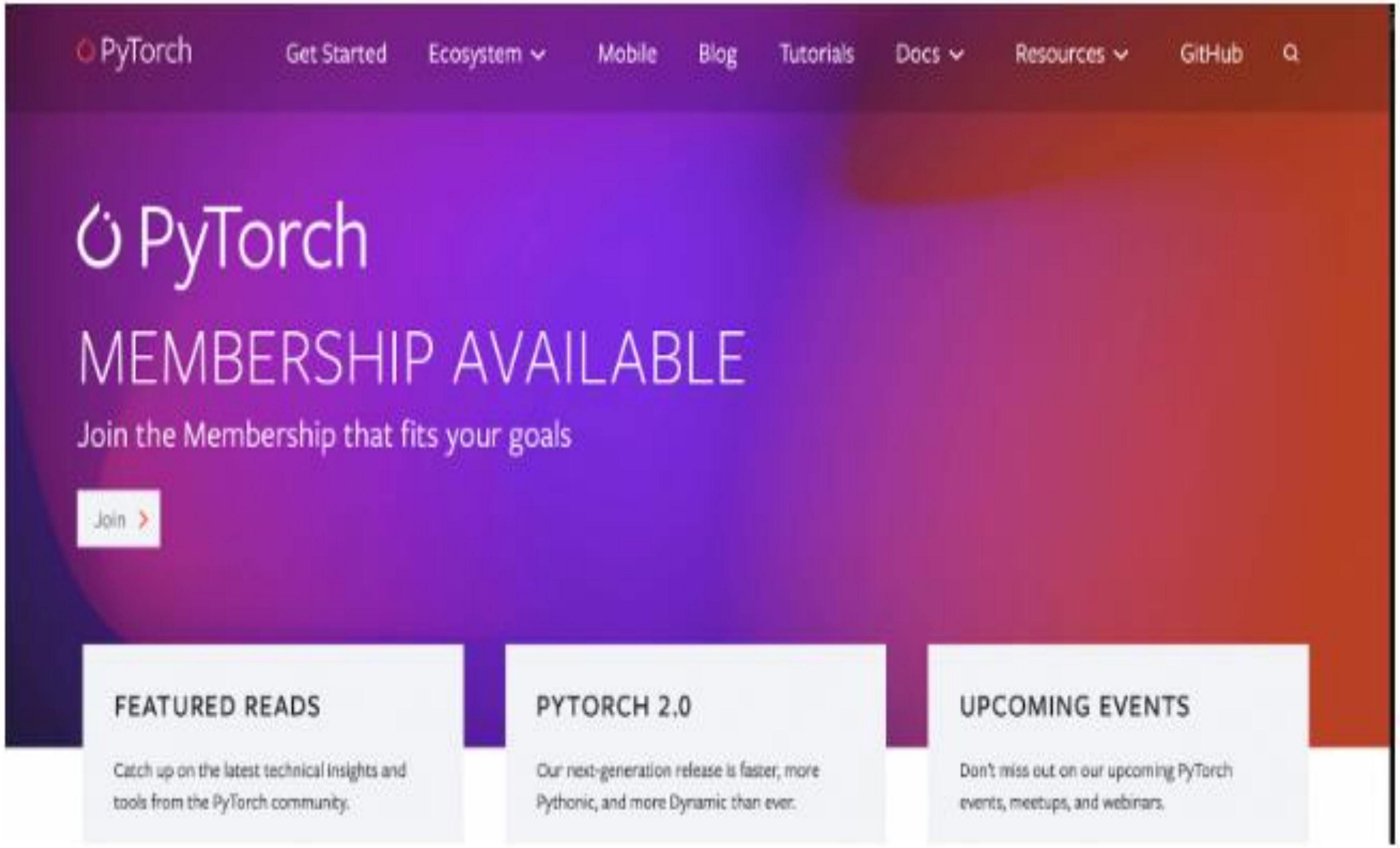
07. Feature Columns

08. Layered Components

09. Event Logger (With TensorBoard)

10. Visualizer (With TensorBoard)

PYTORCH



PyTorch Get Started Ecosystem Mobile Blog Tutorials Docs Resources GitHub

PyTorch

MEMBERSHIP AVAILABLE

Join the Membership that fits your goals

Join

FEATURED READS

Catch up on the latest technical insights and tools from the PyTorch community.

PYTORCH 2.0

Our next-generation release is faster, more Pythonic, and more Dynamic than ever.

UPCOMING EVENTS

Don't miss out on our upcoming PyTorch events, meetups, and webinars.

PyTorch is a dynamic deep learning framework developed by Facebook’s AI Research lab.

It’s favoured for academic research, prototyping, and specific production use cases

With its core built around tensors, PyTorch facilitates building complex neural network architectures and supporting CPU and GPU computation.

Its defining trait is the dynamic computational graph, enabling flexibility in model design and real-time changes.

PyTorch and TensorFlow are similar in that the core components of both are tensors and graphs.

Tensors

Tensors are a core PyTorch data type, similar to a multidimensional array, used to store and manipulate the inputs and outputs of a model, as well as the model’s parameters. Tensors are similar to NumPy’s ndarrays, except that tensors can run on GPUs to accelerate computing.

Graphs

Graphs are data structures consisting of connected nodes (called vertices) and edges. Every modern framework for deep learning is based on the concept of graphs, where Neural Networks are represented as a graph structure of computations. PyTorch keeps a record of tensors and executed operations in a directed acyclic graph (DAG) consisting of Function objects. In this DAG, leaves are the input tensors, roots are the output tensors.

AI Tools for Text Generation:

- GPT-3
- BERT
- OpenAI's Codex:
- Hugging Face
- Talk to TransformerTalk to Transformer

AI Tools for Image Generation:

- Deep Dream Generator
- NVIDIA GauGAN
- Artbreeder
- RunwayML

AI Tools for Video Generation:

- DeepMotion Avatar
- Wibbitz
- Lumen5
- Synthesia

AI Tools for Generating 3D Modules

- SketchUp
- Clara.io
- Blender
- AutoCAD

AI Tools for Coding:

- Kite
- TabNine
- DeepCode
- GitHub Copilot

DEMO ON AI ALGORITHMS USING PYTHON

https://colab.research.google.com/drive/11cpxQXKLzRDrXFaFuBAGJTYp7rE26LO_?authuser=1

https://colab.research.google.com/drive/1esjSlx5dULVa4L4KAo9wPz-aTtqkHq3W

REFERENCES

https://rubikscode.net/2021/08/03/introduction-to-tensorflow-with-python-example/

https://www.tensorflow.org/tutorials

https://www.edureka.co/blog/artificial-intelligence-with-python/

https://www.developer.com/languages/python/ai-with-python/

https://www.tutorialspoint.com/artificial_intelligence_with_python/index.htm

https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3