Artificial Intelligence

AI-2002

Lab 01

Introduction to AI and Python Programming

Course Code: AI-2002	Artificial Intelligence Lab

1. Objective

1. Introduction to Artificial Intelligence, exposing students to different AI Problems and solutions.
2. To acquire programming skills in Python, Introduction to different Python Libraries for Artificial Intelligence.
3. Introduction to different IDE for python programming.
4. Solve some basic AI problem using the python programming language.

2. Introduction to AI

The term Artificial Intelligence was first coined decades ago in the year 1956 by John McCarthy at the Dartmouth conference. He defined AI as: "The science and engineering of making intelligent machines." In other words, Artificial Intelligence is the science of getting machines to think and make decisions like humans. In the recent past, AI has been able to accomplish this by creating machines and robots that have been used in a wide range of fields including healthcare, robotics, marketing, business analytics and many more.

3. Implementation Platform for AI

Artificial Intelligence has been around for over half a century now and its advancements are growing at an exponential rate. The demand for AI is at its peak and this lab course on Artificial Intelligence with Python will help you understand all the concepts of AI with practical implementations in Python. Python has made its way into the most complex technologies such as Artificial Intelligence, Machine Learning, Deep Learning, and so on.

Python is the top programming language in TIOBE and PYPL Index. According to PYPL, which publishes separate ranking for five countries, Python is the top language in all five countries (US,India, Germany, United Kingdom, and France). Python has taken a huge lead in these five countries.

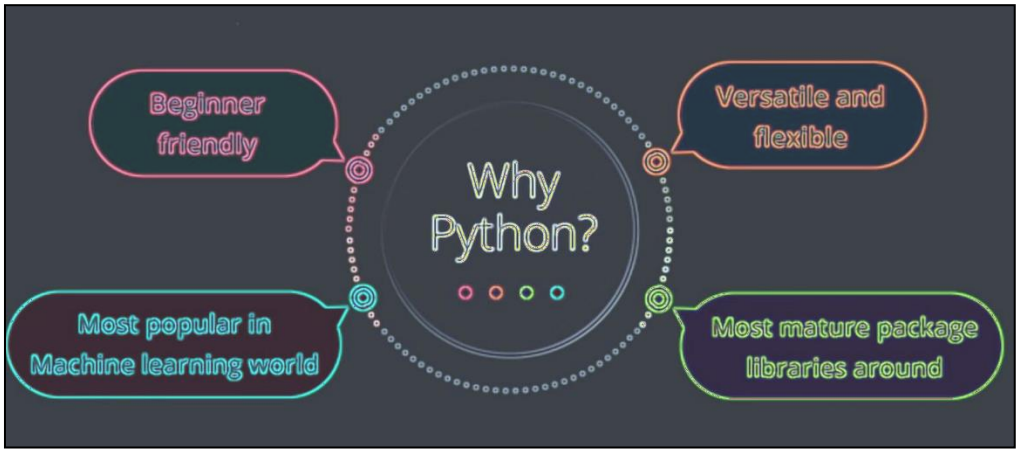
Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		Python	16.36%	+2.78%
2	2		C	16.26%	+3.82%
3	4	^	C++	12.91%	+4.62%
4	3	v	Java	12.21%	+1.55%
5	5		C#	5.73%	+0.05%
6	6		Visual Basic	4.64%	-0.10%
7	7		JavaScript	2.87%	+0.78%
8	9	^	SQL	2.50%	+0.70%
9	8	v	Assembly language	1.60%	-0.25%
10	11	^	PHP	1.39%	-0.00%

3.1 History of Python

- Invented in the Netherlands, early 90s by Guido van Rossum
- Named after Monty Python
- Open sourced from the beginning
- Considered a scripting language, but is much more
- Scalable, object oriented and functional from the beginning
- Used by Google from the beginning
- Increasingly popular

3.2 Distinct Features of Python

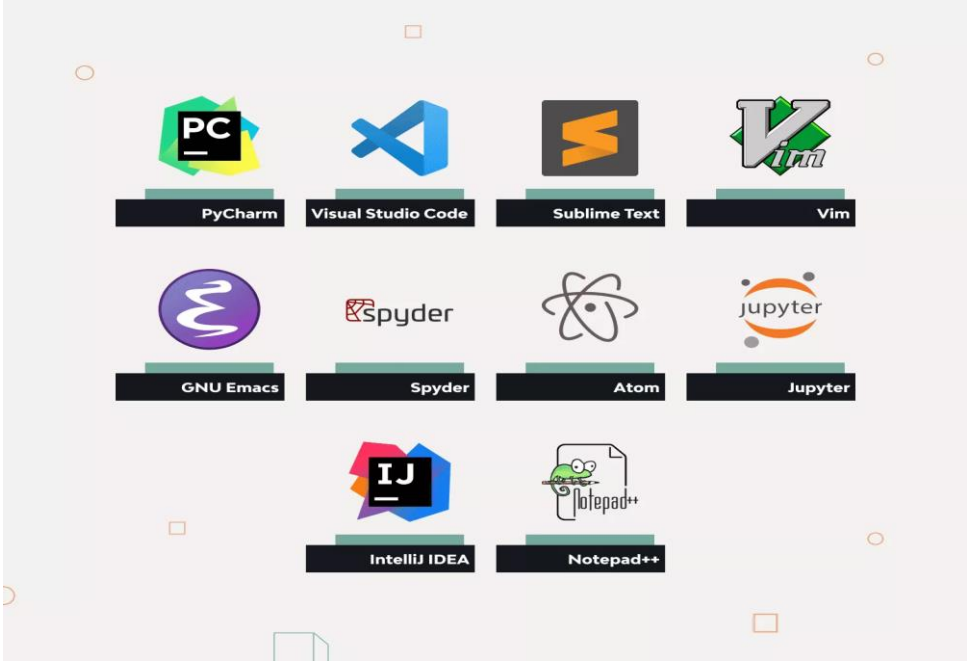
Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, it's not necessary to declare the type of variable because it is a dynamically typed language.



- **Less Code:** Implementing AI involves tons and tons of algorithms. Thanks to Pythons support for pre-defined packages, we don't have to code algorithms. And to make things easier, Python provides "check as you code" methodology that reduces the burden of testing the code.
- **Prebuilt Libraries**: Python has 100s of pre-built libraries to implement various Machine Learning and Deep Learning algorithms. So every time youwant to run an algorithmon a dataset, all you have to do is install and load the necessary packages with a single command. Examples of pre-built libraries include NumPy, Keras, Tensorflow, Pytorch, and so on.
- **Ease of learning:** Python uses a very simple syntax that can be used to implement simple computations like, the addition of two strings to complex processes such as building a Machine Learning model.
- **Platform Independent:** Python can run on multiple platforms including Windows, MacOS, Linux, Unix, and so on.While transferring code fromone platformto the other you can make use of packagessuch as PyInstaller that will take care of any dependency issues.
- **Massive Community Support:** Python has a huge community of users which is always helpful when we encounter coding errors. Apart from a huge fan following, Python has multiple communities, groups, and forums where programmers post their errors and help each other.

3.3 IDE for Python

An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software.



	Features	Spyder	Jupyter Notebook	Google Colab
1	Installation	Yes	Yes	No
2	Launch	GUI	Web browser	Web browser
3	Internet Connection	No	No	Yes
4	Usage	Development	Data Science	Data Science
5	Visualisation	NA	Yes	Yes
6	Data	Local/Remote	Local/Remote	Google Drive
7	Power	CPU	CPU	CPU/GPU/TPU

4. Basic Python Syntax

```
print("Hello, World!")
```

4.1 Assigning Values to Variables

```
a = 2  # a is variable , we are assigning int value 2 to a using Equal to Operator (=)
b= 3.3  # b is variable , we are assigning float value 3.3 to a using Equal to Operator (=)
c= 'xyz'  # c is variable , we are assigning string value 'xyz 'to a using Equal to Operator (=)
```

User Input

```
name=input("enter your name")
print(name)
```

4.3 Basic Data Types

- **Numeric Data Types**

```
int_data = 17

float_data= 12.6

complex_data = 14 + 5j

print('Data : ', int_data , ', Type : ',type(int_data))

print('Data : ', float_data , ', Type : ',type(float_data))
```

```
print('Data : ', complex_data , ', Type : ',type(complex_data))
```

- **Text Data Types**

```
string_data = 'Machine Learning'
string_data_Alphabet = 'a'
string_paragraph = 'Machine learning is a subfield of artificial
intelligence that gives computers the ability to learn without
explicitly being programmed.'
string_num = '12'
print('Data : ', string_data , ', Type : ',type(string_data))
print('Data : ', string_data_Alphabet, ', Type :
',type(string_data_Alphabet))
print('Data : ', string_paragraph , ', Type :
',type(string_paragraph))
print('Data : ', string_num , ', Type : ',type(string_num))
```

Fetching Specific String or complete

```
t = 'Hello World!'

print (t) # Prints complete string

print (t[0])    # Prints first character of the string

print(t[2:5]) # Prints characters starting from 3rd to 5th

print (t[2:]) # Prints string starting from 3rd character

print (t* 2)  # Prints string two times

print (t + "TEST") # Prints concatenated string
```

Data Types to Store Collection

1. List

- Ordered, mutable (changeable), allows duplicates
- Key Features
- Elements are stored in order
- You can modify, add, or remove elements
- Can store different data types

Example

```
my_list = [1, 2, 3, 3, "apple"]
my_list.append(4)
```

Use When

✓ You need an ordered collection

✓ You want to change elements

---

2. Tuple

- Ordered, immutable (cannot be changed), allows duplicates
- Key Features
- Faster than lists
- Values cannot be modified after creation
- Useful for fixed data

Example

```
my_tuple = (1, 2, 3, "apple")
```

Use When

✓ Data should not change

✓ You want data protection

---

3. Set

- Unordered, mutable, does NOT allow duplicates
- Key Features
- Automatically removes duplicates
- Fast membership checking
- No indexing

Example

```
my_set = {1, 2, 3, 3, 4}
# Output: {1, 2, 3, 4}
```

Use When

✓ You want unique values
✓ You need set operations (union, intersection)

---

4. Range

- Ordered, immutable sequence of numbers
- Key Features
- Generates numbers on demand (memory efficient)
- Commonly used in loops

Example

```
numbers = range(1, 6)
```

Use When
✓ You need a sequence of numbers
✓ You want efficient looping

---

5. Dictionary

- Unordered (ordered since Python 3.7), mutable, key-value pairs
- Key Features
- Stores data as key: value
- Keys must be unique and immutable
- Very fast lookups

Example

```
my_dict = {"name": "Riaz", "age": 27}
```

Use When

✓ You want to map keys to values
✓ You need fast data retrieval

Quick Comparison Table

Data Type	Ordered	Mutable	Duplicates	Structure
List	✅	✅	✅	`[ ]`
Tuple	✅	❌	✅	`( )`
Set	❌	✅	❌	`{ }`
Range	✅	❌	❌	`range()`
Dictionary	✅ (3.7+)	✅	Keys ❌	`{key: value}`

4.3 Conditional Statement

If statement

```
age = 18
if age >= 18:
    print("You are eligible to vote.")
```

If Else statement

```
age = 18
if age >= 18:
    print("You are eligible to vote.")
else:
  print("You are not eligible to vote.")
```

Multiple Conditions

```
amount = int(input("Enter amount: "))
if amount<1000:

  discount = amount*0.05
  print("Discount",discount)

elif amount<5000:

  discount = amount*0.10
  print("Discount",discount)

else:

  discount= amount*0.15
  print ("Discount",discount)
  print ("Net payable:",amount-discount)
```

4.4 Loops

For Loop

```
for var in list(range(5)):
  print(var)
#other examples

for letter in 'Python': # traversal of a string sequence
  print ('Current Letter :', letter)

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:  # traversal of List sequence
   print ('Current fruit :', fruit)

print ("Good bye!")
```

While Loop

```
count = 0
while (count < 9):
  print('The count is:', count)
  count = count + 1
  print('Good Bye')
```

4.5 Functions

Non-Parametric Function

```
# Function definition is here
def non_paraFunc ():
  print("This is Non_parametric Function")

non_paraFunc()
```

Parametric Function

```
# Function definition is here
def paraFunc (a,b):
  """This is Non_parametric Function"""
  #Takes the value of a and b parameters provided by the function
user
  return a + b

paraFunc(1,2)
```

Return Statement

```
# Function definition is here
def sum( arg1, arg2 ):
  # Add both the parameters and return them."
  total = arg1 + arg2
  print ("Inside the function : ", total)
  return total

# Now you can call sum function
total = sum( 10, 20 )
print ("Outside the function : ", total )
```

```
# Not returning

def print_message(message):
    print('Print Statement : ', message)
    return  # This is equivalent to return None

result = print_message("Hello, World!")
#We cannot store data if nothing is returned from the statement
print(result)  # Output: None
```

Return Statement

```
# Function definition is here
def sum( arg1, arg2 ):
  # Add both the parameters and return them."
  total = arg1 + arg2
  print ("Inside the function : ", total)
  return total

# Now you can call sum function
total = sum( 10, 20 )
print ("Outside the function : ", total )

# Not returning

def print_message(message):
    print('Print Statement : ', message)
    return  # This is equivalent to return None

result = print_message("Hello, World!")
#We cannot store data if nothing is returned from the statement
print(result)  # Output: None
```

4.5.1 Types of Functions (Based on Arguments)

- Positional
- Default
- Keyword

1. Positional Arguments

Arguments are passed **in the same order** as defined in the function.

Definition

Values are assigned to parameters **by position**.

Example

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

add(2, 3)   # a=2, b=3
```

- Order matters.

---

2. Default Arguments

Parameters are given **default values**, which are used if no argument is passed.

Definition

If a value is not provided, the default value is automatically used.

Example

```
def greet(name="Guest"):
    print("Hello", name)

greet()          # Hello Guest
greet("Alice")   # Hello Alice
```

- Makes arguments optional.

3. Keyword Arguments

Arguments are passed using **parameter names**, so order does not matter.

Definition

Values are assigned by explicitly specifying the parameter name.

Example

```
def display(name, age):
    print(name, age)

display(age=25, name="Alice")
```

- Improves readability and flexibility.

4.5.2 Types of Functions (Based on Number of Parameters)

- Fixed Length
- Variable Length

Fixed Length Parameter

```
#Function with Fixed Parameters
def add(a, b):
  '''A and B are Fixed Parameters'''
  return a + b

add(1,2,3) #If we provide extra arguments, It will return
TypeError
```

Variable Length Parameter

```
# Variable Length Args Argument

def details(*numbers):
  print('numbers : ' ,numbers)

details(1,2,3)

#Keyword Variable Length Args Argument

def details(**name):
  print('Names : ' ,name)

details(n1= 'Xyz', n2= 'Ali', n3= 'Zainab')
```

LAB TASKS

Task 1

Take two numbers from the user.
Store them in variables and:

- Print their sum
- Print their data types

Task 2

Take a string input from the user.
Display:

- First character
- Middle character
- Last character
- Length of the string

Task 3

Create a list containing mixed data types.
Perform the following:

- Replace the second element
- Append a new element
- Remove the last element
- Print the final list

Task 4

Create a tuple of numbers.
Use a loop to:

- Print each element
- Print the total sum of tuple elements

Task 5

Take 10 numbers from the user and store them in a list.
Create a **set** from that list and display:

- Original list
- Set
- Number of duplicate values removed

Task 6

Create a dictionary to store student details:

- name
- marks
- grade

Using conditions:

- Assign grade based on marks
- Update the dictionary and print it

Task 7

Write a program that:

- Takes a number from the user
- Prints whether it is **Prime or Not** using a loop

Task 8

Using `range()` and a `for` loop:

- Print all even numbers between 1 and 50
- Count how many even numbers are printed

Task 9

Write a program using a `while` loop that:

- Takes a number from the user
- Prints its multiplication table
- Stops when the multiplier reaches 10

Task 10

Create a function that:

- Accepts variable-length arguments
- Returns the **maximum** number

Create another function that:

- Accepts keyword arguments
- Prints them in `key : value` format