KEY POINTS

- instarling Python & VS code
- old course = but good

NAME/DATE/SUBJECT

Introduction to Programming

Python Lecture 1 23-6-2025

NOTES

Why python ⇒ Very versatile

⇒ Can be used in AI ML

Programming :-

Just like we use Hindi or English to Communicate with each others, we use a programming language like python to Communicate with the Computer.

Programming is a way to instruct the Computer to perform various tasks

Q. what is python?

python is a Simple and easy to learn Language which feels like reading simple english.

This psuedo Code nature of python makes it easy to learn & understandable by begimers

Features of python

Easy to understand = less developed time

Free & open Source

High level Language

portable → works on Linux/windows/Mac

+ fun to work with

installation

python Can be easily installed from python.org

when you click on the download button. python Can be installed right after you Completed the setup by executing the file on your platform

just install it like a game

SUMMARY

Python is good & useful

download it

KEY POINTS

- python & pycharm
- Installed pycharm
- Next class is main

NAME/DATE/SUBJECT

Downloading python and Pycharm installation

Python lecture 2 23-6-2021

NOTES

pycharm helps to run python

like a code editor like vs code

From now on we use pycharm instead of VS Code

SUMMARY

KEY POINTS

- pip & Modules

To install a module
Pip install ____

NAME/DATE/SUBJECT

Modules & pip

Python lecture 3 25-6-25

NOTES :-

Modules :- It is a code which was already written by others and we can use it

To write that code we may take a lot of time so we can use that

To install module
Pip install ____

we don't need to memorize all the modules

we can use chatgpt or google to get inbuilt modules

What is pip?

Pip is like a package manager of python
↳ it will pull out modules

Types of Module
1. Built in module → those modules are present in Python itself → no need of internet
2. external modules → have to get from internet

SUMMARY

Pip install ____

KEY POINTS

Syntax of the function: print

print("_____")

NAME/DATE/SUBJECT

Writing our first python Programme → "Hello world"

Python Lecture 4 25-6-25

NOTES

The first python programme

open pycharm & use

the function "print"

NOTE:- Don't name your file as pyradio or time as available module names because it can lead to Confusion & Trouble if we use that module in our file

So we will be writing our code in pycharm for now

There is a parameter that we should write print in the first programme

Syntax of print

print("_____")

After installing the module we can restart it

Now important notes from Harry

Let's write our very first python programme.

Create a file and use the function print ("You Know python") ← So when you save this file you will see the output below

Modules:- A module is a file containing Code written by Somebody else(Generally) which Can be imported & used in our programs

Pip:- pip is the "package manager" for python

You Can use pip to install a module from your system.

You already know Syntax for pip

SUMMARY

KEY POINTS

Using python as Calculator

NAME/DATE/SUBJECT

Using python as Calculator

Python lecture 5

27-6-25

NOTES

In this video Harry showed how to use python as calculator,

We simply open the powershell (or) Terminal then Type "python"

after that perform Calculations

Notes :-

we can use python as a Calculator by typing "python +↵ on the terminal

↳ This opens REPL

Read, Evaluate, Point Loop

SUMMARY

KEY POINTS

1. use , for multi print
2. # and ''' ''' for comments
3. escape sequences
   (\n \t \' \" \\)
4. use of end=(" ")

NAME/DATE/SUBJECT

Comments, Escape sequences & Print statement

Python Lec 6 27-6-25

NOTES

In this video Harry showed Comments, Escape sequences, chars, and print function features

Comments

Syntax:- # _____ → single line Comment

''' _____ → Multi line Comment

use ctrl+/ to Comment Completely and anywhere

Def:- Comments are used to write something which the programmer does not want to execute

↓ Can be used to mark author name, date etc.

Other features of print

We know syntax of print

eg Print("This is a note")
Print("python note")
} then output is like
This is a note
python note

eg Print("This is a note", end=" ")
Print("python note")
} Then output is like
This is a notepython note

So this, end=" " does is it will make another print statement start in the same line

if we want to use multiple statements at once in print then use ,(comma)

Escape Sequence characters

eg Print("C:/Himesh") ⇒ C:/Himesh
Print("c:\himesh") ⇒ c
himesh

This is because in python \ backslash denotes for escape characters

imp escape characters to learn

\n → new line
\t → new tab
\' → Single quote
\" → Double quote
\\ → backslash

} To use this write the code

SUMMARY

→ use escape sequences for required stuff

→ while doing multiple print in same function use ,

eg Print("This is a note", "Python note")
↓ This is a note Python note
if you use comma space will come

eg Print("This is a note", end=" ")
Print("Python note")
↓ This is a notePython note
no space if you use end

KEY POINTS

→ Variables ← int, string, float

for Conversion use typeCastors like: str() int() float()

Strings Concatenate
eg "31" + "45" = "3145"
to get 76 convert them to int

Input function

```
Input ( )
```

NAME/DATE/SUBJECT

Variables, Data types & Type Casting

Python lecture 7 part 1 28-6-25

NOTES

Variable:-

A Variable is a name given to a memory location in a programme. For eg

a = 30 b = "Himesh" c = 7122

→ Variables = Container to store a value

→ Keywords = Reserved words in python

→ Identifiers = class/function/ Variable name

Datatypes

Primarily there are following data types in python

1. Integers
2. Floating point numbers
3. Strings
4. Booleans
5. None

python is a fantastic language that automatically identifies the type of data for us:

a = 71 ⇒ identifies a as class<int>

b = 88.44 ⇒ identifies b as class<float>

name = "Himesh" ⇒ identifies name as class<str>

Rules for defining a Variable name= also applies to other identifiers

→ A variable name can contain alphabets, digits & underscores

→ A variable name can only start with an alphabet and underscore

→ A variable name cant start with a digit

→ No space is allowed to be used inside a variable name.

Examples of few variable names are:

Himesh, one8, seven_, _seven

Type function & Type Casting

* Type function is used to find the datatype of a given variable in python eg a = 31 type(a) ⇒ class<int> b = "31" type(b) ⇒ class<str>

A number can be Converted into a string and vice versa (if possible)

There are many functions to convert one data type into another

str(31) = "31" ⇒ Integer to string Conversion

int("32") = 32 ⇒ String to integer Conversion

float(32) = 32.0 ⇒ integer to float Conversion ... And so on-

SUMMARY

The datatypes & their Conversion

→ input function

```
input ( )
```

KEY POINTS

NAME/DATE/SUBJECT

Variables, Data types & Type Casting

Python lecture 7 part2 28-6-25

NOTES

Input( ) function

This function allows the user to take input from the keyboard as a "String"

⇒ if a is "himesh" then user entered himesh

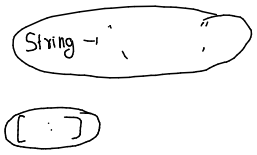
eg a = input("Enter your name")

⇒ if a is "34" then user entered 34

It is important to note that the output of input is always a string (even if the number is entered

SUMMARY

KEY POINTS



NAME/DATE/SUBJECT

→ Strings & Slicing
→ Advanced Slicing techniques

Python Lec-8 Part 1 29-6-25

NOTES

Strings:-

KB if we write [—:—] even if we type a no greater than length → we get output till length only

→ String is a datatype in python

→ String is a seq of characters enclosed in quotes

eg str = "Himesh"
Print(str[2:897897])
↳ mesh

→ We can Primarily, write a string in three ways

1. Single quoted strings ⇒ a = 'Himesh'

2. Double quoted Strings ⇒ b = "Himesh"

3. Triple Quoted string → c = """ Himesh """

also if we just print the string in a bar ie

Str: "Himesh"
Print(str[2]) → Then it will print the character at that digit place

String slicing

A String in python can be sliced for getting a part of the String

Consider the following String:-

Name = H i m e s h ⇒ length = 5
(-6 -5 -4 -3 -2 -1 / 0 1 2 3 4 5)

If there is a function called as len() which tells us the length of the string

The index in a string starts from 0 to (length-1) in python. In order to slice a string we use the following syntax

S1 = Name [ind_start : ind_end]
first index included; last index not included

S1[0:3] returns Him ⇒ characters from 0 to 3

S1[1:3] returns im ⇒ characters from 1 to 3

Negative indices:- Negative indices can also be used as shown in the fig above. -1 corresponds to length-1 index, similarly for others

INP INFO KB → First convert negative slicing into positive slicing

Slicing with skip value:-

We can provide a skip value to string as a part of our slice like this

Word = "Amazing"

Word [1:6:2] → mzn

Other advance slicing techniques

Word = "Amazing"
Word [::-1] → Word[0:7:-1] → gnizamA
Word [0::-1] → Word[0:0:-1] → nizamA

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT #String functions

Python lec 8- part 2

NOTES

String Functions

Some of the mostly used functions to perform operation on or manipulate strings are

1. len() function :→ This function returns the length of the string

len("Himesh") → returns 6

2. string.ends with ("—")→ This function tells whether the variable ends with the string"—" or not. If String is "Himesh" it returns true for "esh" since Himesh ends with "esh"

3. string.Count ("i") → Counts the total no. of occurance of any character

4. String.capitalize ( ) → This function Capitalize the first character of a given String

5. String.find (word) → This function finds a word & returns the index of first occurence of that word in the String.

6. String.replace (old word, new word) → This function replaces the old word with new words in the entire string.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Python Lists & list fxdions

Python lecture 9 part 1 — 2-7-2025

NOTES

Python Lists:-

python lists are containers to store a set of values of any data type

eg:-

friends = ["Apple", "Akash", "Rohan", 7, False]
(str) (int) (Boolean)

Can store value of any datatype

List Indexing:-

A list can be indexed like a string

L1 = [7, 9, "Himesh"]

L1[0] ⇒ 7 L1[1] ⇒ 9 L1[10] ⇒ error: L1[0:2] ⇒ [7,9]
(List slicing)

List slicing always returns a list.

This slicing is similar to String

Warning: Dont use negative indices skip value ie [_:_:-1]
Don't execeed this by number -1

List Methods:-

Consider the following list

L1 = [1, 9, 7, 2, 21, 15]

① L1.sort() → Sorts the lists in ascending order (Only if all are int or floating pnt)

② L1.reverse() → reverses the list order

③ L1.append(8) → add the told element in the end of the list

④ L1.insert(_, _) ⇒ This will add that in that index & return its value
(at index, add this)

⑤ L1.pop(2) → will delete element at index 2

⑥ L1.remove(_) → will remove that element from the list

SUMMARY

List & list functions

KEY POINTS

NAME/DATE/SUBJECT #Tuples

Python Lec 9 part 2

NOTES

Tuples in Python :-

A tuple is an immutable data type in python
↳ Can not change

a = ( ) ⇒ empty tuple

a = (1,) ⇒ Tuple with a single element requires a comma

eg a = (1) ⟹ print output 1
a = (1,) ⟹ print output (1,)

a = (1, 7, 2) ⇒ A Tuple with more than one element

once defined a tuple's elements Cant be altered or manipulated

Tuple methods :-

Consider the following tuple

a = (1, 7, 2)

1. a.count (1) : a.count() will return number of times 1 occurs in a.
2. a.index(1) : a.index() will return the index of first occurence of 1 in a.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Dictionary & its functions

Python lec 10

3-7-25

NOTES

Dictionary:- Dictionary is a collection of key-value pairs

Syntax: {"key": "value"

eg {"himesh": "Code",

"marks": "100",

"List": [1,2,3]}

a["key"] ⇒ prints "value"

a["list"] ⇒ prints [1,2,3]

Properties of a python Dictonaries:-

1. It is unordered
2. It is mutable
3. It is indexed
4. Cannot contain duplicate keys.

Dictionary Methods

Consider the following dictionary

a = {"name": "Himesh",

"from": "Indo",

"marks": [92,98,96]

Pradeep: {M:Dosa, A:Idli, N:Pizza}}

① a.items() : returns a list of (Key, value) tuples

② a.keys() : returns a list containing dictionary's keys

③ a.update({"friend": "Sam"}) : updates the dictionary with supplied key-value pairs

④ a.get("name") : returns the value of specified keys (and value is returned eg: "Himesh" is returned here

For more methods Use AI

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

#Apni Dictionary

Python Lec 11 : 5-7-25

NOTES

a Create a dictionary and take input from the user and return the meaning of the word from the dictionary

SUMMARY

KEY POINTS

1 Syntax of sets

$S = set(L)$

L ⇒ List

$L = [a, b, c, f]$

NAME/DATE/SUBJECT #Sets, in python

Python Lecture 12 5-7-2025

NOTES

Sets in python:-

Set is a collection of non repetitive elements

S = Set ( ) ⇒ No repetition is allowed

S. add (1) ⇒ or Set = {1,2}

S. add (2)

If you are a programming beginner without much knowledge of mathematical operations on sets, you can simply look at sets in python as data types containing unique values

Properties of Sets:-

1. Sets are unordered ⇒ elements orders doesn't matter
2. Sets are unindexed ⇒ we cant access elements by index
3. There is no way to change items in sets
4. Sets cant contain duplicate values

Operations on sets:-

Consider the following set:

S = {1, 8, 2, 3}

1. len(S) : Returns the length of the set, Here 4
2. S.remove(x) : updates the sets and remove the x from the set
3. S.pop( ) : Removes an arbitary element from the set and returns the element removed

eg S = set(1,2,3,4)

S.pop( ) ⇒ output is {1,3,4}

4. S.clear( ) : Empties the set
5. S.union(S_2) ⇒ simply union of 2 sets $S \& S_2$
6. S.intersection(S_2) ⇒ intersection b/w 2 sets $S \& S_2$

SUMMARY

KEY POINTS

imp

Syntax:

```
if(Condition):
    Print("Yes")        # if Condition 1 is true
elif(Condition 2):
    Print("No")         # if Condition 2 is true
else:
    Print("Maybe")      # otherwise
```

NAME/DATE/SUBJECT

#if else conditionals

Python lecture 13 — 7-7-2025

NOTES

Sometimes we want to play BGMI on our phone if the day is sunday

Sometimes we order Icecream online if the day is sunny.

Sometimes we go hiking if our parents allow

All these are decisions which depends on a Condition being met.

In python programming too, we must be able to execute instructions on a condition(s) being met

This is what Conditionals are for

If, else & elif in python

If else & elif statements are a multiway decision taken by our program due to Certain Condition in our Code

imp

Syntax:

```
if(Condition):
    Print("Yes")        # if Condition 1 is true
elif(Condition 2):
    Print("No")         # if Condition 2 is true
else:
    Print("Maybe")      # otherwise
```

We can also use if in place of elif like this

```
Var1 = 23
Var2 = 76
var3 = int(input())
if Var3 > Var2:
    Print("O")
if Var3 == Var2:
    Print("X")
else:
    Print("ईगा")
```

problem Here is that our code have to efficient i.e if के बाद अगर if true है तो उधर काम होजाता अगर elif use करे तो पर अगर if use करे तब वो दुबारा read करने लगेगा

Code example:

```
a = 22
if (a>9):
    Print("Greater")
else:
    Print("lesser")
```

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Relational operators

#Python lecture 14: 10-7-2025

NOTES

Relational operators:-

Relational operations are used to evaluate conditions inside the if statements.

Some examples are -

== ⇒ equal

>= ⇒ greater than/equal to

<=

logical operators:-

In python logical operators operate on conditional statements -eg

and ⇒ true if both operands r true else false

or ⇒ true if at least one operand is true else false

not ⇒ inverts true to flase & false to true

elif clause:-

elif in python means [else if]. An if statements can be chained together with a lot of these elif statements followed by an else statements

```
if (Condition):
    #code

elif (Condition 2):
    #code
```

⇒ This ladder will stop

```
elif (Condition3):
    #code
```

once a condition in an if or elif is met

```
else:
    #code
```

Important notes:-

1. There can be any number of elif statements
2. Last else is executed only if all the conditions inside elifs fail

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

For loops:-

python lecture 16: Loops 13-7-2023

NOTES

Loops In Python!

Sometimes we want to repeat a set of statements in our program. for instance: Print 1 to 1000

Loops make it easy for a programmer to tell the Computers which set of instructions to repeat and how!

Types of loops in python:-

Primarily there are two types of loops in python:-

1. while loop
2. For loop

We will look into these one by one!

While loop

The Syntax of a while loop looks like this:

```
While(Condition):      => The block keeps executing
    # Body of the loop until the Condition is true
```

In while loops, the Condition is checked first. If it evaluates to true, the body of loop is executed, otherwise not!

If the loop is entered, the process of [Condition check & execution] is continued until the Condition becomes false.

Q: Quick Quiz: Write a programme to print 1 to 50 using a while loop An eg: i = 0

Note: If the Condition never becomes False, the loop keeps getting executed

```
While i<5 :
    print ("Harry")
    i = i+1
```

=> "Harry" "Harry" "Harry" "Harry" "Harry"

Prints "Harry" 5 times!

Q. Quick Quiz: write a statement programme to print the Content of a list using while loops.

For loop:-

A for loop is used to iterate through a Sequence like lists, tuple or string [iterables]

The Syntax of a for loop looks like this:

```
L = [ , , , ]

for item in L:
    Print(item)  => 1,7,88
```

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

#Break & Continue

Python Lecture 17: Break & Continue 14-7-2025

NOTES

For loop with else if

An optimal else can be used with a for loop if the code is to be executed when the loop exhaxhausted.

Eg:-

```
l=[1,7,8]

for item in l:
        Print(item)
else: print ('done')   # This is printed when the loop exhausts
```

1 7 8 Done

The Break statement:-

'Break' is used to come out of the loop when encountered. It instructs the program to Exit the loop now

```
eg: for i in range (0, 20):
        Print(i)
    if i==3:          # This will print 0,1,2 and 3.
        Break
```

The Continue statement:-

"Continue" is used to stop the interaction of the loop and Continue with the next one.

It instructs the progamm to "Skip the iteration"

eg:

```
for i in range(10):
    Print("printing")
    if i==2           # if i=2 then iteration is skipped
      Continue.
    Print(i)
```

eg: i = 0

```
while(True):
    if i<5:
        i=i+1
        Continue
    else
        print("Yo!")
```

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Python lecture 18: Functions & doc strings 15-7-2025

NOTES

Function:-

A function is a group of statements performing a specific task.

When a program gets bigger in size and its complexity grows, it gets bigger in size and its complexit grows, it gets difficult for a programmer to keep track on which piece of code is doing what!

A function can be reused by the programmer in a given program any no. of times.

Example & syntax of a function:-

The syntax of a function looks as follows:

```
def name():
    print("Hello")
```

This function can be called any ,10 or times, anywhere in the program.

Function call:-

Whenever we want to call a function, we put the name of the function followed by parenthesis as follows:

name() → This is called a function call

Function definition-

The part containing the exact set of instructions. which are executed during the function call.

Quick quiz: write a programme to greet a user with "Good day" using functions.

Types of functions in python:-

There are two types of functions in python:

1. Built in functions → Already present in Python
2. user defined functions → Defined by the user

Eg of but in functions includes len(), print() etc

The function func(), we defined is an example of user defined function

Functions with arguments

A function can accept some values it can work with, we can put these values in the parenthesis. A function can also return as shown below:

```
def greet(name):
    gr = "Hello" + name
    return gr
a = greet("Himesh") → Hello Himesh
```

SUMMARY

Doc string:- It is like a multi line comment which describes that function, the first string of a function

to get doc string:-

print(f.__doc__)

KEY POINTS

NAME/DATE/SUBJECT

Lecture 19:-

16-7-25

Python Lecture 19: Try exception handling, File Io Basics

NOTES

Try Exception handling.

Sometimes in our code for some reason we can expect a error but after that error

we may have imp code to run eg: when net is off in YT that is an err but we dont want to completely shut down the app so, we will print Something.

Syntax:-

lets go by Example :-

```
num1 = input("Enter num1", )
num2 = input("Enter num2", )

Print("Sum of these are", int(num1)+int(num2))→error
Print("Karansuno")→will not come!
```

P1 ↑ P2 ↓

∴ we try this

```
try:
    P1→ command
Here
→ P2 (Here)
```

File Io Complete notes Later! ie Tmrw

SUMMARY

KEY POINTS

Q. What is file

→ Generally memory in Computer → Hard drive → non volatile
→ RAM → Volatile

The memory stored in non-volatile format is called file.

Different modes by which you can open a file

"r" → open file for reading default
"w" → open file for writing
"x" → Creates file if not exists
"a" → Append Add more Content to a file
"t" → text mode default
"b" → binary mode
"+" → Read write it update

"rb" → will open for read in binary mode
"rt" → will open for read in text mode

NAME/DATE/SUBJECT

Python lecture 20: File Io 19-7-25

NOTES

The Random Access memory is volatile and all its Contents are lost once a programme terminates

In order to persist forever the data is Stored as files.

A file is data stored in a storage device. A python program can talk to the File by reading Content from it and writing Content to it.

RAM = Volatile, SSD/HDD → Non volatile

Types of files:-

There are 2 types of files:

1) Text files (.txt, .c etc)

2) Binary files (.jpg, .dat etc)

python has a lot of functions for reading, updating and deleting files.

Opening a file:-

Python has an open() function for opening files. It takes 2 parameters: file name and mode

open("this.txt", "r")
open is a built-in function
Filename
mode of opening (read mode)

Reading a file in python

f = open("this.txt", "r") → open the file in r mode

text = f.read() → Read its contents

Print(text) → print its Contents

f.close() → close the file

We can also Specify the number of characters in read() function: f.read(2)
↳ Read first 2 characters

other methods to read the file

① f.readline()

→ We can use f.read line() to read on full line at a time

f.readline() → Reads one line from the file

② f.readlines()

→ This will make Convert the file into a list with each line as elements of list

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

19-7-25

Python lecture 21: Write & append in File IO

NOTES

Writing files in python:-

In order to write to a file, we first open it in write or append mode after which we use the python's f.write() method to write to the file

```
f=open("this.txt","w")
f.write("this is nice")
f.close()
```

↳ can be called multiple times

if we want to add text to the file without erasing its data we use "a"

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

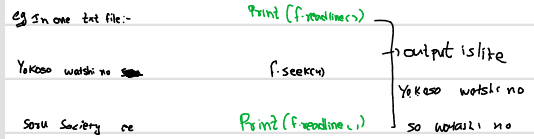
20-7-2025

Python lecture 22: Seek and tell in files I/o.

NOTES

Seek function:-

f.seek() It will reset the file pointer and will bring it from our input



ie पूरे file को उस number character से पढ़ना start करेगा!

Tell function:-

Tell() → This will tell the no. of characters in the line

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Python lecture 23: With blocks 20-7-25

NOTES

#Syntax:-

```
with open(" ") as f :
```

This does the something which f=open() and f.close does

ie no need to write f.close now

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

20-7-2025

Python lecture 24 : Scope, Global variables and Global Keyword :-

NOTES

→ A variable outside of the Function is called Global variable

→ A variable inside the function is Called Local variable

Eg: L=10 → Global

```
def func(n):
    L = 5 → Local
    Print (L)
```

Will first look for Local variable if not there then goes for Global variable even after that if variable not found then throws an error

Global Keyword

Normally in the function we cannot change the Global variable but by using this we can change.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

23-7-25

Python lec 25: Recursions :-

NOTES

Recursions:-

Recursion is a function which calls itself.

It is used to directly use a mathematical formula as a function for eg:

$n! = n(n-1)!$

This function can be defined as follows:

```
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)
```

Base Condition which doesn't call the function any further

Function calling itself

This work as follows:

Factorial(4)
↓
[Function called]

4 x Factoial(3)
↓
4 x [3 x Factorial(2)]
↓
4 x [3 x 2[factorial(1)]]
↓
4 x 3 x 2 x [1] [Function returned]

The programmer need to be extremely careful while working with recursion to ensure that the function doesn't infinitely keep calling itself.

Recursion is sometimes the most direct way to code an algorithm

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

13-8-2025

Python lecture 26: Anonymous/lambda fnc

NOTES

Q. What is a lambda function?

→ A lambda function is a small, anonymous function defined using the lambda keyword

→ Syntax:

lambda argument : expression

#Key characteristics:

- No Name (anonymus)
- Single expression only (no multiple statements)

→ Used for short, throw-away functions.

like using functions without creating function

Eg:-

```
# Square of a number
Square = lambda x : x**2
# Add two numbers:
Sum = lambda a,b: a+b
```

#Common use cases:

- with map( ):- Apply a function to all items in a list.

```
list(map(lambda x : x**2 ,[1,2,3]))
```

- with filter( ): Filter items based on a condition

```
list(filter(lambda x : x%2==0, [1,2,3,4]))
```

#When to use:-

- for quick inline functions
- When defining a full function with def unnecessary.

#Limitations:-

→ Not suitable for complex logic

→ less readable for beginners!

SUMMARY

KEY POINTS

Top modules in AI & M

1. Numerical Computing: Numpy
   Fast array operation, Linear algebra.
2. Scientific Computing: Scipy
   Advanced math, optimisation, Signal processing.
3. Data Handling: Pandas.
   Dataframes, cleaning, aggregation.
4. Big Data: Dask, Vaex
   Scaleable Data processing.
5. Machine learning: Scikit-learn, PyCaret
   ML algorithm, Module evaluation
6. Deep learning: Tensorflow, Py Torch.
   Neural networks, GPU acceleration
7. NLP: Spacy, NLTK, Textblob
   Text processing, Sentiment analysis
8. Transformers: Hugging face Transformers:
   Pretrained model for NLP, vision, audio
9. Visualization: Matplotlib, Seaborn
   charts, plots, Statistical graphs
10. OCR: Tesseract, Pytesseract
    Text extraction from images.
11. Web Scraping: Beautiful Soup, Scrapy
    Extract data from websites
12. Data Engineering: Airflow, Luigi:
    workflow automation

NAME/DATE/SUBJECT

14-08-2025

Python lecture 27: more about modules.

NOTES

Q. what are modules.

- Modules are reuseable python files Containing functions, classes or variables.
- They help to organize code and avoid repetition.

Types of Modules:-

Built-in Modules:- Come with python eg. math, random, datetime

External Modules:- installed via pip (eg requests, numpy)

importing modules:- → import generally
→ using pip

Benefits:-

- Speeds up development
- Reduces bugs by using tested Code
- Enable access to advanced functionality.

Basic modules for me:

S No	module name	function.
1	math	Basic operations: sqrt(), pow(), radians()
2	random	Random number generation:- randint(), choice()
3	datetime	Date/time handling: datetime.now(), timedelta()
4	os	interact with os: listdir(), remove(), getCwd()
5	sys	System specific parameters: argv, exit()
6	re	Regular expressions: search(), findall()
7	Statistics	Datastats: mean(), median(), mode()
8	turtle	Simple graphics for fun learning: forward(), circle()

Random:-

random.randint() → gives a random int

random.choice() gives a random element in that inputed group

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

F-Strings:

Python lecture 28: 14-08-2025

NOTES

Q. What are F-strings?

- Syntax: f"{}"

Why use F-strings:-

- Cleaner & more readable
- faster, Supports inline expressions and function calls().

Basic usage:-

```
name = "Himesh"
age = "17"
Print(f"Hi my name is {name} and Im {age} old")
```

inline Expression:-

```
Print(f"Next Year i will be {age +1} Years old")
```

function Call

```
Print(f"Name length: {len(name)}")
```

Formatting numbers.

```
Pi = 3.14159
Print(f"Rounded: {Pi:.2f}") →output 3.14
```

Text Allignment-

```
Print(f"{name:<10}") # Left allign in 10 spaces
Print(f"{name:>10}") # Right allign.
```

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Args & Kwags:

Python lecture 29: 16-08-2025

NOTES

Purpose of args & Kwargs:

- *args → positional Arguments → Tuple
- **Kwargs → Keyword arguments → Dictionary

KB these features are like optional

*Args

→ Used when you want to pass multiple values to a function

→ These values are captured as a tuple.

eg: def sum_all(*args):
    return Sum(args)

Sum_all(1, 2, 3) => output 6

**Kwargs:-

→ Used when you want to pass named arguments.

→ These are stored as dictionary

eg: def print_info(**kwargs):
    for key, value in kwargs.items():
        Print(f"{key}:{value}")

Print_info(name = "Himesh", age = 17) => Himesh:17

if Some def function as def func(*args, normal) we will get an err!

Rule → always first normal, *args, **Kwargs.

What each parts mean:

Normal: A standard required argument (eg string or number)

*args: A tuple of extra positional arguments.

**kwargs: A dictionary of extra keyword arguments.

Q Why args & Kwargs?

→ if in a func there are 4 arguments but we wanted to pass 5 or 6 arguments there.

To resolve that problem we use args & Kwargs

KB we don't always need to type *args only we can type *anything (args) or **anything (kwargs)

But this is standard

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Time modules:-

Python lecture 30: 16-8-2025

NOTES

Time module:-

The time module in python allows you to work with time-related functions such as delays, timestamps, and performance measurements

Key functions:-

① time.time()

- Returns current time in seconds, minutes, hours
- used to measure execution time

```
import time

start = time.time()

#Your code

end = time.time()

Print(end - start)
```

2. time.sleep(seconds)

- Pauses execution for given seconds
- Useful for delays in loops or automation

```
time.sleep(2) # waits for 2 seconds.
```

3. time.localtime()

- Converts epoch time to readable structure

```
print(time.localtime())
```

4. time.strftime(format, time_object)

```
formatted = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
```

5. time.getnow()

- fetches current time

Performance Comparision example

Used time.time() to compare execution time of:

- for loop
- while loop

[helps in benchmarking & optimizing code]

Use Cases:

- Game development
- Automation scripts
- logging timestamps
- Performance benchmarking

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Python lecture 31:- Virtual environment:-

17-8-2025

NOTES

Virtual environment:- what is it?

→ A virtual environment is a self-contained directory that Contains a python installation & its dependencies.

it allows you to:

- isolate project-specific packages.
- Avoid conflicts b/w diff projects
- Maintain clean & manageable setups.

Q. Why use it.

Imagine working on two diff python projects:

- Project A needs Django==3.2 } installing both would cause conflicts
- project B needs Django==4.0 } A virtual environment solves this by keeping dependencies separate.

Q. How to Create a virtual environment.

→ Open cmd prompt in project folder (used chatGpt for a problem which occured.

→ Run:

```
Python -m venv myenv
```

- my env is the name of your environment folder.

3. Activate the environment:

```
myenv\Scripts\activate
```

- You'll see (my env) in your terminal indicating it's active.

4. Deactive when done:

```
deactivate
```

Installing packages inside virtual Environment:-

Once activated, use pip as usual:

```
pip install numpy
```

→ This installs numpy only inside myenv, not globally.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

requirements.txt

Python lecture 32. 17-8-2025

NOTES

what is it?

→ A requirements.txt files stores all the packages your project needs it helps:

- Share dependencies with teammates.
- Recreate environment easily.
- Deploy projects with consistent set-ups.

How to Create it

Pip freeze > requirements.txt

This saves all installed packages with versions

Examples Content:

numpy == 1.24.0

request == 2.31.0

How to use it

To install packages from the file:

pip install -r requirements.txt

Real-word use case

Let's say you're building a web app:

1. Create a virtual environment
2. install Flask and other packages.
3. Save dependencies to requirements.txt
4. Share your project with a friend.
5. They run pip install -r requirements.txt and get the same setup.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Enumerate function:-

Python lecture 33. 18-8-2025

NOTES

Q. What is enumerate()?

enumerate() is a built in python function that helps you to loop through items in a list (or any iterable) and gives you both item and its index.

Syntax:-

Enumerate(iterable, start=0)

iterable: The list or string you want to loop through

Start:. The starting index (default is 0)

eg: Fruits=["apple","banana","ichigo")

```
for index, fruit in enumrate (fruits):
    Print(index, fruit)
```

0. apple
1. banana
2. ichigo

#To start Counting from anothes number:-

```
for i, fruit in enumrate(fruits,start=1):
    Print(i, fruit)
```

1. apple
2. banana
3. ichigo

Q Why use enumrate()?

→ You get both index and value easily.

→ no need to Create a Separate Counter.

→ Make your Code cleaner and easier to read.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

More about import

Python lecture 34 18-08-2025

NOTES

Q. What is import?

The import statement allows you to use code from other python files or libraries called modules) in your program

Types of modules
- External: download from net
- Built in: already there.
- Custom modules:- Your own .py files with reuseable code

Import Syntax:-

Version 1: Normal:-	version 2: selective import	Version 3: Rename module
import math	from math import sqrt	import math as m
Print(math.sqrt(16))	Print(sqrt(16))	Print(m.sqrt(25))

Importing Custom modules:-

1. Create a python file (eg., mytools.py)
2. Define functions inside it
3. import it:

```
import mytools
mytools.greet()
```

Q. How python finds modules:-

Python Searches in this order:

1. Current working directory
2. installed packages
3. System paths listed in sys.path

```
import sys
Print(sys.path)
```

Tips:
① Avoid circular imports
② Keep module name unique
③ Use __name__ = "__main__" to prevent unwanted execution.

Useful Tools:-

- dir(module) → lists all functions/variables in a module
- help(module) → shows documentation

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

19-8-2025

Python lecture 35:

__name__ = __main__

Usage & necessity

NOTES

Q. What is __name__?

· __name__ is a special in built variable in python.

· when a python file is run directly, __name__ is set to "__main__"

· when the same file is imported as a module, __name__ becomes the modules name.

Q. Why use if __name__ == "__main__"?

· it ensures that certain code runs only when the file is executed directly, not when imported

· prevents the unintended execution of code during module imports.

· Commonly used to wrap test code, main logic, or script entry points.

Real life analogy

imagine a movie script: some scenes should only play when the move is the main feature - not when it is part of a trailer.

→ Similarly, if __name__ == "__main__" acts like a gatekeeper for your script's main content

· Code example:-

```
def greet():
    Print("Hello From greet()")
if __name__ == "__main__":
    greet()
```

· If this file is run directly → output: Hello from greet()

· If imported → no output unless greet() is called explicitly.

Benefits in project structure

→ makes your code modular and reuseable

→ Encourages clean separation b/w logic and execution.

→ Essential for collaborative workflows and larger projects.

Best practice:-

· Always wrap executable code in if __name__ == "__main__" block.

· Keep reuseable functions and classes outside this block.

· Helps in writing unit tests and debugging modules safely.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Join function.

Python lecture 36: 20-8-2025

NOTES

Purpose:

Join() is used to combine elements of an iterable (like a list or tuple) into a single string using a specified separator.

Syntax:-

Separator.join(iterable)

Separator: A string that will be placed b/w each element.

iterable: A sequence (like list, tuple) containing only strings.

eg:

```
words=["Python","is","awesome"]
result = " ".join(words)        # Python is awesome
Print(result)
```

→Common use cases:-

Joining words into sentences

Creating csv-style strings

→Efficient string concatenation in loops.

→Important rules:-

- All elements in the iterable must be strings.

```
nums=[1,2,3]
','.join(map(str,nums))   #Convert to strings first
```

→Join is faster than using + in loops for string concatenation

Quick tip:

Use map(str, iterable) to safely convert non-string items before joining.

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Map, filter & Reduce:-

Python lecture 37: 20-8-2025

NOTES

1. map(): Apply a function to all Elements

Purpose: Transforms each item in an iterable using a function.

Syntax: (wo func jo tum chahte ho ki puree list par apply ho!)

map(function, iterable)

eg: def sq(x):

    return x*x

numbers = [1,2,3]

result = list(map(sq,ware, numbers)) → [1,4,9]

☑ use it when you want to modify every element.

2. filter() - Select Specific elements

Purpose: Filters items based on a Condition.

Syntax:

filter(function, iterable)

eg:

def is-even(x):

    retun x%2==0

numbers = [1,2,3,4]

result = list(filter(is--even, numbers)) ⇒ [2,4]

☑ use it when you want to keep only matching items.

3. reduce(): Combine All elements into one.

Purpose: Applies a function cumulatively to reduce iterable to a single value

Syntax:

from functools import reduce

reduce(function, iterable)

eg: def add(x,y):

    return x+y

numbers=[1,2,3,4]

result = reduce(add, numbers) → 10

☑ use it when you want to aggregate values.

Quick tips:

- Use lambda for short anonymous func:

  list(map(lambda x: x*2, [1,2,3])) → [2,4,6]

- map() & filter() return iterators → wrap with list() to view results.

- reduce() is not a built in → import from functools

SUMMARY

KEY POINTS

NAME/DATE/SUBJECT

Decorators in Python :-

Python lecture 38. 21-08-2025

NOTES

Q. What is a decorator?

A decorator is a function that takes another function as input and extends or modifies its behaviour without changing it's source code

eg:
```
def my_decorator(func):
    def wrapper():
        Print("Before function runs")
        func()
        Print("After function runs")
    return wrapper

@my_decorator
def say_hello():
    Print("Hello!")

say_hello()
```

output:

Before function runs

Hello!

After function runs

Key Concepts:

- First-class functions: Functions can be passed as arguments, returned and assigned to variables.
- @decorator_name Syntax: A shorthand to apply decorators
- Wrapper function: inner function that adds extra behavior.

Use Cases:

- Logging . Access control . Timing execution . Caching . Validation

Advanced Decors:-

- With arguments: Decorators that accept parameters.
- Class-based decorators: Use __call__() method.
- Chaining decorators: Apply multiple decorators to a single function.

SUMMARY