

ULTIMATE PYTHON HANDBOOK



YOUR COMPLETE GUIDE TO
PYTHON PROGRAMMING

```
# CodeWithHarry
def learn():
    print("Code.")
    print("Practice.")
    print("Build.")
    return True

for skill in ["Learn", "Practice", "Build"]:
    print(skill)

if __name__ == "__main__":
    learn()
    print("Keep Growing!")
```



BEGINNER
TO ADVANCED



PRACTICAL
EXAMPLES



REAL-WORLD
PROJECTS



BEST
PRACTICES

CODE. PRACTICE. BUILD. REPEAT.

Python Programming Handbook

CHAPTER 01

Beginner Friendly Learning Guide

What is Programming?

Just like we use Hindi or English to communicate with each other, we use programming languages like Python to communicate with computers.

Programming is the process of giving instructions to a computer so it can perform tasks, solve problems, and automate work.

What is Python?

Python is one of the easiest and most beginner friendly programming languages. Its syntax feels close to simple English which makes it easy to read, write, and understand.

Python is widely used in web development, automation, artificial intelligence, machine learning, data science, and software development.

Features of Python

01

Easy to learn and beginner friendly syntax.

02

Free and open source programming language.

03

Works on Windows, Linux, and Mac systems.

04

Widely used in modern technologies and AI.

Installation

GETTING STARTED

Just install it like a game

Python can be downloaded from python.org Download the installer for your operating system and complete the setup process.



Python Programming Handbook

CHAPTER 01

Beginner Friendly Learning Guide

Modules, Comments & Pip

Let's write our very first python program. Create a file called hello.py and paste the below code in it.

```
print("hello world") # print is a function
```

Execute this file by typing python hello.py and you will see Hello World printed on the screen.

Modules

A module is a file containing code written by somebody else which can be imported and used in our programs.

Pip

Pip is the package manager for Python. You can use pip to install external modules on your system.

```
pip install flask
```

Types of Modules

There are two types of modules in Python.

01

Builtin Modules are preinstalled with Python.

02

External Modules need to be installed using pip.

Examples of builtin modules are os and random. Examples of external modules are tensorflow and flask.



Python as a Calculator

REPL MODE

Use Python directly from the terminal

We can use Python as a calculator by typing python in the terminal. This opens REPL which stands for Read Evaluate Print Loop.



Python Programming Handbook

CHAPTER 01

Beginner Friendly Learning Guide

Comments

Comments are used to write something which the programmer does not want to execute. They can be used to explain code, mark author names, dates, or add notes.

Types of Comments

There are two types of comments in Python.

Single Line Comments

To write a single line comment, add a # symbol at the start of the line.

```
# This is a Single-Line Comment
```

Multiline Comments

To write multiline comments, you can use triple quotes.

```
"""  
This is an amazing  
example of a Multiline  
comment!  
"""
```



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 01

Practice Set

1. Write a program to print Twinkle twinkle little star poem in python.
2. Use REPL and print the table of 5 using it.
3. Install an external module and use it to perform an operation of your interest.
4. Write a python program to print the contents of a directory using the os module. Search online for the function which does that.
5. Label the program written in problem 4 with comments.



Python Programming Handbook

CHAPTER 02

Beginner Friendly Learning Guide

Variables and Datatype

A variable is the name given to a memory location in a program. For example.

```
a = 30      # variables = container to store a value
b = "harry" # keywords = reserved words in python
c = 71.22   # identifiers = class/function/variable name
```

Data Types

Primarily these are the 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 = "harry" # identifies name as class <str>
```

Rules for Choosing an Identifier

- A variable name can contain alphabets, digits, and underscores.
- A variable name can only start with an alphabet and underscores.
- A variable name can't start with a digit.
- No white space is allowed to be used inside a variable name.

Examples of a few variable names are: harry, one8, seven_, _seven etc.

Operators in Python

Following are some common operators in python:

1. Arithmetic operators: +, -, *, / etc.
2. Assignment operators: =, +=, -= etc.
3. Comparison operators: ==, >, >=, <, != etc.
4. Logical operators: and, or, not.

Python Programming Handbook

CHAPTER 02

Beginner Friendly Learning Guide

type() Function and Typecasting

type() function is used to find the data type of a given variable in python.

```
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)    # integer to string conversion
int("32")  # string to integer conversion
float(32)  # integer to float conversion
```

... and so on.

Here "31" is a string literal and 31 a numeric literal.

Input() Function

This function allows the user to take input from the keyboard as a string.

```
a = input("enter name") # if a is "harry", the user entered harry
```

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



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 02

Practice Set

1. Write a python program to add two numbers.
2. Write a python program to find remainder when a number is divided by z.
3. Check the type of variable assigned using input() function.
4. Use comparison operator to find out whether 'a' given variable is greater than 'b' or not.
Take a = 34 and b = 80
5. Write a python program to find an average of two numbers entered by the user.
6. Write a python program to calculate the square of a number entered by the user.



Python Programming Handbook

CHAPTER 03

Beginner Friendly Learning Guide

Strings

String is a data type in python.

String is a sequence of characters enclosed in quotes.

We can primarily write a string in these three ways.

```
a = 'harry'      # Single quoted string
b = "harry"      # Double quoted string
c = '''harry'''   # Triple quoted string
```

String Slicing

A string in python can be sliced for getting a part of the strings.

Consider the following string:



Positive index starts from 0 (left to right)

Negative index starts from -1 (right to left)

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

s = name [start : end]

first index **included**

last index is **not included**

Python Programming Handbook

CHAPTER 03

Beginner Friendly Learning Guide

Slicing with Skip Value

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

```
word = "amazing"

word[1:6:2] # mzn
```

Other advanced slicing techniques:

```
word = "amazing"

word[-7:-1] # amazin
word[:7]    # amazing
word[0:]    # amazing
```

String Functions

Some commonly used functions to manipulate strings are:

1. `len()` returns the length of the string.

```
str = "harry"
print(len(str)) # Output: 5
```

2. `endswith()` checks if a string ends with given text.

```
str = "harry"
print(str.endswith("rry")) # Output: True
```

3. `count()` counts total occurrences of a character.

```
str = "harry"
count = str.count("r")
print(count) # Output: 2
```

4. `capitalize()` capitalizes the first character.

```
str = "harry"
capitalized = str.capitalize()
print(capitalized) # Output: Harry
```


Python Programming Handbook

CHAPTER 03

Beginner Friendly Learning Guide

String Functions Continued

5. `find()` returns the index of first occurrence.

```
str = "harry"  
index = str.find("rr")  
print(index) # Output: 2
```

6. `replace(old word, new word)` replaces the old word with the new word in the string.

```
str = "harry"  
replaced = str.replace("r", "l")  
print(replaced) # Output: hally
```

Escape Sequence Characters

Sequence of characters after backslash `"\"` are called Escape Sequence characters.

These characters represent one special character inside strings.

Example:

`\n`

`\t`

`\'`

`\\`

etc.

`\n`

newline

line 1



line 2

`\t`

Tab

move cursor
forward

A ■ ■ ■ ■ B

`\'`

single quote

'

`\\`

backslash

\\

Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 03

Practice Set

1. Write a python program to display a user entered name followed by Good Afternoon using input() function.
2. Write a program to fill in a letter template given below with name and date.

```
letter = '''
Dear <|Name|>,
You are selected!
<|Date|>
'''
```

3. Write a program to detect double space in a string.
4. Replace the double space from problem 3 with single spaces.
5. Write a program to format the following letter using escape sequence characters.

```
letter = "Dear Harry, this python course is nice. Thanks!"
```



Python Programming Handbook

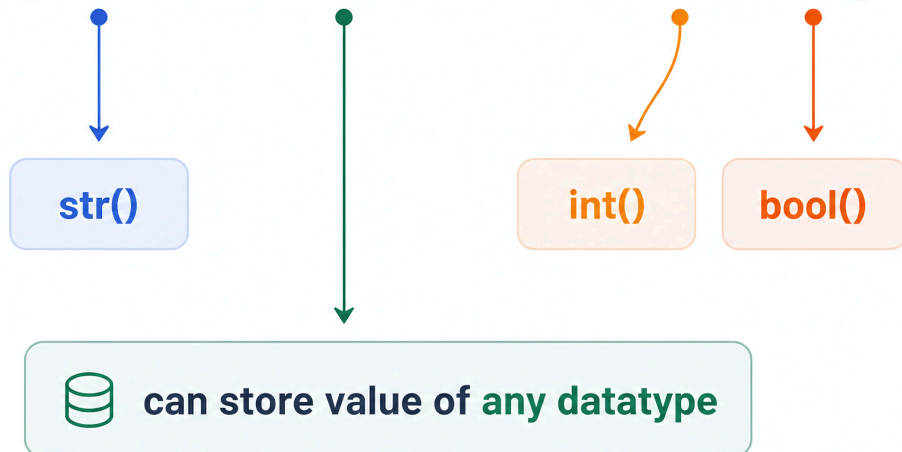
CHAPTER 04

Beginner Friendly Learning Guide

Lists And Tuples

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

```
friends = ["apple", "akash", "rohan", 7, false]
```



List Indexing

A list can be indexed just like a string.

```
l1 = [7, 9, "harry"]

l1[0] # 7
l1[1] # 9
l1[2] # harry

l1[0:2] # [7, 9] # list slicing
```

List Methods

Consider the following list.

```
l1 = [1, 8, 7, 2, 21, 15]
```

- `l1.sort()`: updates the list to `[1, 2, 7, 8, 15, 21]`
- `l1.reverse()`: updates the list to `[15, 21, 2, 7, 8, 1]`
- `l1.append(8)`: adds 8 at the end of the list
- `l1.insert(3, 8)`: This will add 8 at 3 index

Python Programming Handbook

CHAPTER 04

Beginner Friendly Learning Guide

- `l1.pop(2)`: Will delete element at index 2 and return its value.
- `l1.remove(21)`: Will remove 21 from the list.

Tuples In Python

A tuple is an immutable data type in python.

```
a = () # empty tuple

a = (1,) # tuple with only one element needs a comma

a = (1,7,2) # tuple with more than one element
```

Tuple Methods

Consider the following tuple.

```
a = (1, 7, 2)
```

- `a.count(1)`: a count (1) will return number of times 1 occurs in a.
- `a.index(1)` will return the index of first occurrence of 1 in a.



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 04

Practice Set

1. Write a program to store seven fruits in a list entered by the user.
2. Write a program to accept marks of 6 students and display them in a sorted manner.
3. Check that a tuple type cannot be changed in python.
4. Write a program to sum a list with 4 numbers.
5. Write a program to count the number of zeros in the following tuple:

```
a = (7, 0, 8, 0, 0, 9)
```



Python Programming Handbook

CHAPTER 05

Beginner Friendly Learning Guide

Dictionary & Sets

Dictionary is a collection of keys-value pairs.

Syntax:

```
a = {  
    "key": "value",  
    "harry": "code",  
    "marks": "100",  
    "list": [1, 2, 9]  
}  
  
print(a["key"]) # Output: "value"  
print(a["list"]) # Output: [1, 2, 9]
```

Properties Of Python Dictionaries

- It is unordered.
- It is mutable.
- It is indexed.
- Cannot contain duplicate keys.

Dictionary Methods

Consider the following dictionary.

```
a={"name": "harry",  
   "from": "india",  
   "marks": [92, 98, 96]}
```

- `a.items()`: Returns a list of (key,value) tuples.
- `a.keys()`: Returns a list containing dictionary's keys.
- `a.update({"friends":})`: Updates the dictionary with supplied key-value pairs.
- `a.get("name")`: Returns the value of the specified keys.

More methods are available on docs.python.org



Python Programming Handbook

CHAPTER 05

Beginner Friendly Learning Guide

Sets In Python

Set is a collection of non-repetitive elements.

```
s = set() # no repetition allowed!

s.add(1)

s.add(2) # or set ={1,2}
```

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

- Sets are unordered ⇒ **Element's order doesn't matter**
- Sets are unindexed ⇒ **Cannot access elements by index**
- There is no way to change items in sets.
- Sets cannot contain duplicate values.

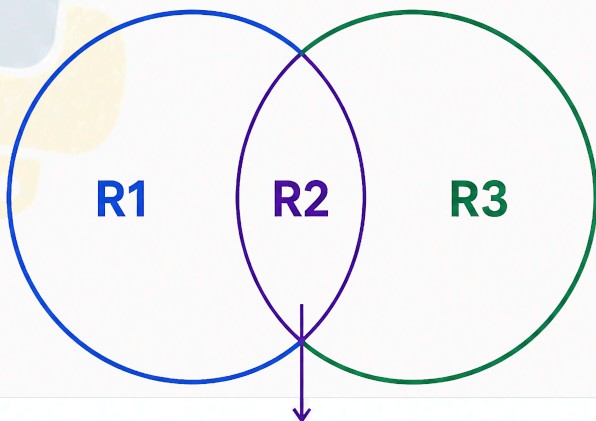
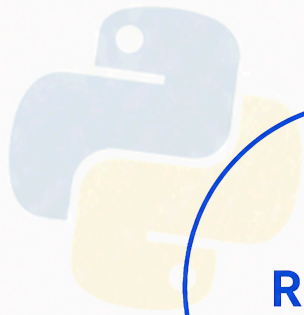
Operations On Sets

Consider the following set:

```
s = {1, 8, 2, 3}
```

- `len(s)`: Returns 4, the length of the set
- `s.remove(8)`: Updates the set `s` and removes 8 from `s`.
- `s.pop()`: Removes an arbitrary element from the set and return the element removed.
- `s.clear()`: empties the set `s`.
- `s.union({8,11})`: Returns a new set with all items from both sets.
- `s.intersection({8,11})`: Returns a set which contains only item in both sets {8}.





A **$A \cap B$** **B**

$$R2 = A \cap B$$

$$R1 + R2 + R3 = A \cup B$$

$$R1 + R3 = A \Delta B$$

$$R1 = A - B$$

$$R3 = B - A$$



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 05

Practice Set

1. Write a program to create a dictionary of Hindi words with values as their English translation. Provide user with an option to look it up!
2. Write a program to input eight numbers from the user and display all the unique numbers (once).
3. Can we have a set with 18 (int) and '18' (str) as a value in it?
4. What will be the length of following set s:

```
s = set()
s.add(20)
s.add(20.0)
s.add('20') # length of s after these operations?
```

5. `s = {}`

What is the type of 's'?

6. Create an empty dictionary. Allow 4 friends to enter their favorite language as value and use key as their names. Assume that the names are unique.
7. If the names of 2 friends are same; what will happen to the program in problem 6?
8. If languages of two friends are same; what will happen to the program in problem 6?
9. Can you change the values inside a list which is contained in set S?

```
s = {8, 7, 12, "Harry", [1, 2]}
```



Python Programming Handbook

CHAPTER 06

Beginner Friendly Learning Guide

Conditional Expression

Sometimes we want to play PUBG on our phone if the day is Sunday.

Sometimes we order Ice Cream online if the day is sunny.

Sometimes we go hiking if our parents allow.

All these are decisions which depend 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 And Elif In Python

If else and elif statements are a multiway decision taken by our program due to certain conditions in our code.

Syntax:

```
if(condition1): # if condition1 is True
    print("yes")

elif(condition2): # if condition2 is True
    print("no")

else:           # otherwise
    print("maybe")
```

Code Example.

```
a=22

if(a>9):
    print("greater")

else:
    print("lesser")
```

Quick Quiz: Write a program to print yes when the age entered by the user is greater than or equal to 18.

Python Programming Handbook

CHAPTER 06

Beginner Friendly Learning Guide

Relational Operators

Relational Operators are used to evaluate conditions inside the if statements. Some examples of relational operators are:

`==`: equals.

`>=`: greater than/ equal to.

`<=`: lesser than/ equal to.

Logical Operators

In python logical operators operate on conditional statements. For Example:

- `and` – true if both operands are true else false.
- `or` – true if at least one operand is true or else false.
- `not` – inverts true to false & 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` statement.

```
if (condition1):  
    #code  
  
elif (condition2):    # this ladder will stop once a condition in an if or elif is met.  
    #code  
  
elif(condition3):  
    #code  
  
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.



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 06

Practice Set

1. Write a program to find the greatest of four numbers entered by the user.
2. Write a program to find out whether a student has passed or failed if it requires a total of 40% and at least 33% in each subject to pass. Assume 3 subjects and take marks as an input from the user.
3. A spam comment is defined as a text containing following keywords: "Make a lot of money", "buy now", "subscribe this", "click this". Write a program to detect these spams.
4. Write a program to find whether a given username contains less than 10 characters or not.
5. Write a program which finds out whether a given name is present in a list or not.
6. Write a program to calculate the grade of a student from his marks from the following scheme:

90 – 100 ⇒ Ex
80 – 90 ⇒ A
70 – 80 ⇒ B
60 – 70 ⇒ C
50 – 60 ⇒ D
<50 ⇒ F
7. Write a program to find out whether a given post is talking about "Harry" or not.



Python Programming Handbook

CHAPTER 07

Beginner Friendly Learning Guide

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 computer which set of instructions to repeat and how!

Types Of Loops In Python

Primarily there are two types of loops in python.

- while loops
- for loops

We will look into these one by one.

While Loop

Syntax:

```
while (condition): # The block keeps executing until the condition is true
    #Body of the loop
```

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

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

Quick Quiz: Write a program to print 1 to 50 using a while loop.

Example:

```
i = 0

while i < 5: # print "Harry" - 5 times!
    print("Harry")
    i = i + 1
```

Note: If the condition never become false, the loop keeps getting executed.

Quick Quiz: Write a program to print the content of a list using while loops.



Python Programming Handbook

CHAPTER 07

Beginner Friendly Learning Guide

For Loop

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

Syntax:

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

for item in l:
    print(item) # prints 1, 7 and 8
```

Range Function In Python

The range() function in python is used to generate a sequence of number.

We can also specify the start, stop and step-size as follows:

```
range(start, stop, step_size)
# step_size is usually not used with range()
```

An Example Demonstrating Range() Function.

```
for i in range(0,7): # range(7) can also be used

    print(i) # prints 0 to 6
```

For Loop With Else

An optional else can be used with a for loop if the code is to be executed when the loops exhausts.

Example:

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

for item in l:
    print(item)

else:
    print("done") # this is printed when the loop exhausts!
```

Python Programming Handbook

CHAPTER 07

Beginner Friendly Learning Guide

The Break Statement

'break' is used to come out of the loop when encountered. It instructs the program to – exit the loop now.

Example:

```
for i in range(0,80):  
    print(i)      # this will print 0,1,2 and 3  
  
    if i==3:  
        break
```

The Continue Statement

'continue' is used to stop the current iteration of the loop and continue with the next one. It instructs the Program to "skip this iteration".

Example:

```
for i in range(4):  
    print("printing")  
  
    if i == 2: # if i is 2, the iteration is skipped  
        continue  
  
    print(i)
```

Pass Statement

pass is a null statement in python.

It instructs to "do nothing".

Example:

```
l = [1,7,8]  
  
for item in l:  
    pass      # without pass, the program will throw an error
```


Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 07

Practice Set

1. Write a program to print multiplication table of a given number using for loop.
2. Write a program to greet all the person names stored in a list 'l' and which starts with S.

```
l = ["Harry", "Soham", "Sachin", "Rahul"]
```

3. Attempt problem 1 using while loop.
4. Write a program to find whether a given number is prime or not.
5. Write a program to find the sum of first n natural numbers using while loop.
6. Write a program to calculate the factorial of a given number using for loop.
7. Write a program to print the following star pattern.

```
*  
***  
***** for n = 3
```

8. Write a program to print the following star pattern:

```
*  
**  
*** for n = 3
```

9. Write a program to print the following star pattern.

```
* * *  
* * for n = 3  
* * *
```

10. Write a program to print multiplication table of n using for loops in reversed order.



Python Programming Handbook

CHAPTER 08

Beginner Friendly Learning Guide

Functions & Recursions

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

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

A function can be reused by the programmer in a given program any number of times.

Example And Syntax Of A Function

The syntax of a function looks as follows:

```
def func1():  
    print('hello')
```

This function can be called any number of times, anywhere in the program.

Function Call

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

```
func1() # This is called function call.
```

Function Definition

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

Quick Quiz: Write a program to greet a user with "Good day" using functions.

Types Of Functions In Python

There are two types of functions in python:

- Built in functions (Already present in python)
- User defined functions (Defined by the user)

Examples of built in functions includes len(), print(), range() etc.

The func1() function we defined is an example of user defined function.



Python Programming Handbook

CHAPTER 08

Beginner Friendly Learning Guide

Functions With Arguments

A function can accept some value it can work with. We can put these values in the parentheses.

A function can also return value as shown below:

```
def greet(name):  
    gr = "hello"+ name  
    return gr  
  
a = greet ("harry")  
#a will now contain "hello harry"
```

Default Parameter Value

We can have a value as default as default argument in a function.

If we specify name = "stranger" in the line containing def, this value is used when no argument is passed.

Example:

```
def greet(name = "stranger"):  
    # function body  
  
greet() # name will be "stranger" in function body (default)  
  
greet("harry") # name will be "harry" in function body (passed)
```

Recursion

Recursion is a function which calls itself.

It is used to directly use a mathematical formula as function.

Example:

```
factorial(n) = n x factorial (n-1)
```

This function can be defined as follows:

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

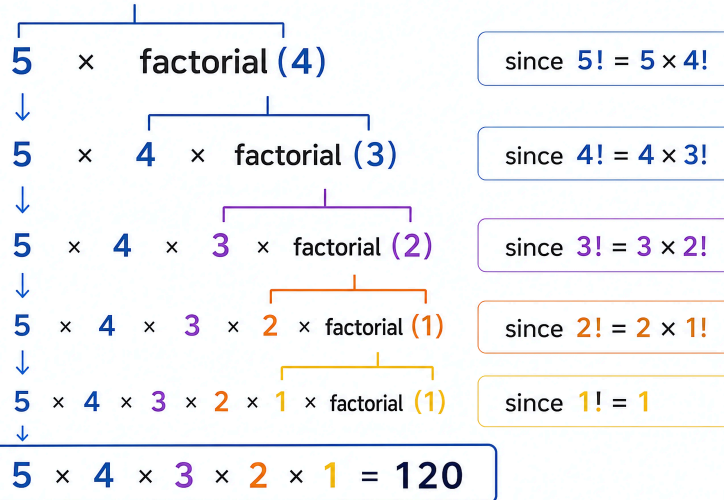

Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 08

This works as follows:

FACTORIAL (5)



The programmer needs 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.



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 08

Practice Set

1. Write a program using functions to find greatest of three numbers.
2. Write a python program using function to convert Celsius to Fahrenheit.
3. How do you prevent a python print() function to print a new line at the end.
4. Write a recursive function to calculate the sum of first n natural numbers.
5. Write a python function to print first n lines of the following pattern.

```
***  
**      - for n = 3  
*
```
6. Write a python function which converts inches to cms.
7. Write a python function to remove a given word from a list and strip it at the same time.
8. Write a python function to print multiplication table of a given number.



Python Programming Handbook

Beginner Friendly Learning Guide

Project

Project 1 : Snake Water Gun Game

We all have played snake, water gun game in our childhood. If you haven't, google the rules of this game and write a python program capable of playing this game with the user.



Python Programming Handbook

CHAPTER 09

Beginner Friendly Learning Guide

File I/O

The random-access memory is volatile, and all its contents are lost once a program terminates. In order to persist the data forever, we use 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.

Programmer



Computer Program
written in Python

Write

Read

FILE

RAM = Volatile

HDD = Non Volatile

Type 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: filename and mode.

```
# open("filename", "mode of opening(read mode by default)")
open("this.txt", "r")
```



Reading A File In Python

```
# Open the file in read mode
f = open("this.txt", "r")

# Read its contents
text = f.read()

# Print its contents
print(text)

# Close the file
f.close()
```

Other Methods To Read The File.

We can also use `f.readline()` function to read one full line at a time.

```
f.readline() # Read one line from the file.
```

Modes Of Opening A File

r – open for reading

w – open for writing

a – open for appending

+ – open for updating.

'rb' will open for read in binary mode.

'rt' will open for read in text mode.



Python Programming Handbook

CHAPTER 09

Beginner Friendly Learning Guide

Write 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!

```
# Open the file in write mode
f = open("this.txt", "w")

# Write a string to the file
f.write("this is nice")

# Close the file
f.close()
```

With Statement

The best way to open and close the file automatically is the with statement.

```
# Open the file in read mode using 'with'

with open("this.txt", "r") as f:

    # Read the contents of the file
    text = f.read()

# Print the contents
print(text)
```



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 09

Practice Set

1. Write a program to read the text from a given file 'poems.txt' and find out whether it contains the word 'twinkle'.
2. The game() function in a program lets a user play a game and returns the score as an integer. You need to read a file 'Hi-score.txt' which is either blank or contains the previous Hi-score. You need to write a program to update the Hi-score whenever the game() function breaks the Hi-score.
3. Write a program to generate multiplication tables from 2 to 20 and write it to the different files. Place these files in a folder for a 13-year-old.
4. A file contains a word "Donkey" multiple times. You need to write a program which replaces this word with ##### by updating the same file.
5. Repeat program 4 for a list of such words to be censored.
6. Write a program to mine a log file and find out whether it contains 'python'.
7. Write a program to find out the line number where python is present from ques 6.
8. Write a program to make a copy of a text file "this.txt".
9. Write a program to find out whether a file is identical and matches the content of another file.
10. Write a program to wipe out the content of a file using python.
11. Write a python program to rename a file to "renamed_by_python.txt".



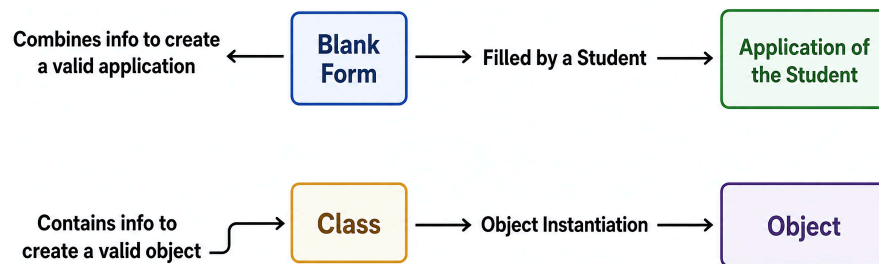
Object Oriented Programming

Solving a problem by creating object is one of the most popular approaches in programming. This is called object-oriented programming.

This concept focuses on using reusable code (DRY Principle).

Class

A class is a blueprint for creating object.



Syntax:

```
class Employee: # Class name is written in pascal case

    # Methods & Variables
```

Object

An object is an instantiation of a class. When class is defined, a template (info) is defined. Memory is allocated only after object instantiation.

Objects of a given class can invoke the methods available to it without revealing the implementation details to the user. – **Abstractions & Encapsulation!**

Modelling A Problem In OOPS

We identify the following in our problem.

- Noun → **Class** → **Employee**
- Adjective → **Attributes** → **name, age, salary**
- Verbs → **Methods** → **getSalary(), increment()**



Python Programming Handbook

CHAPTER 10

Beginner Friendly Learning Guide

Class Attributes

An attribute that belongs to the class rather than a particular object.

Example:

```
class Employee:
    company = "Google" # Specific to Each Class

harry = Employee() # Object Instantiation

harry.company

Employee.company = "YouTube" # Changing Class Attribute
```

Instance Attributes

An attribute that belongs to the Instance (object). Assuming the class from the previous example:

```
harry.name = "harry"
harry.salary = "30k" # Adding instance attribute
```

Note: Instance attributes take preference over class attributes during assignment & retrieval.

When looking up for harry.attribute it checks for the following:

- 1) Is attribute present in object?
- 2) Is attribute present in class?

Self Parameter

self refers to the instance of the class. It is automatically passed with a function call from an object.

```
harry.getSalary() # here self is harry

# equivalent to Employee.getSalary(harry)
```

The function getSalary() is defined as:

```
class Employee:
    company = "Google"

    def getSalary(self):
        print("Salary is not there")
```


Static Method

Sometimes we need a function that does not use the self-parameter. We can define a static method like this:

```
@staticmethod # decorator to mark greet as a static method

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

__init__() Constructor

__init__() is a special method which is first run as soon as the object is created.

__init__() method is also known as constructor.

It takes 'self' argument and can also take further arguments.

For Example:

```
class Employee:

    def __init__(self, name):
        self.name=name

    def getSalary(self):
        ...

harry = Employee("Harry")
```



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 10

Practice Set

1. Create a class *"Programmer"* for storing information of few programmers working at Microsoft.
2. Write a class *"Calculator"* capable of finding square, cube and square root of a number.
3. Create a class with a class attribute `a`; create an object from it and set `'a'` directly using `'object.a = 0'`. Does this change the class attribute?
4. Add a static method in problem 2, to greet the user with hello.
5. Write a Class `'Train'` which has methods to book a ticket, get status (no of seats) and get fare information of train running under Indian Railways.
6. Can you change the self-parameter inside a class to something else (say *"harry"*)? Try changing self to *"slf"* or *"harry"* and see the effects.



Python Programming Handbook

CHAPTER 11

Beginner Friendly Learning Guide

Inheritance & More On OOPS

Inheritance is a way of creating a new class from an existing class.

Syntax:

```
class Employee: # Base class
    # Code

class Programmer(Employee): # Derived or child class
    # Code
```

We can use the method and attributes of 'Employee' in 'Programmer' object.

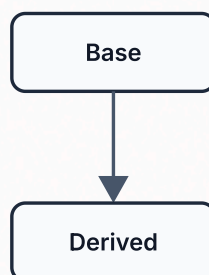
Also, we can overwrite or add new attributes and methods in 'Programmer' class.

Types Of Inheritance

- Single inheritance
- Multiple inheritance
- Multilevel inheritance

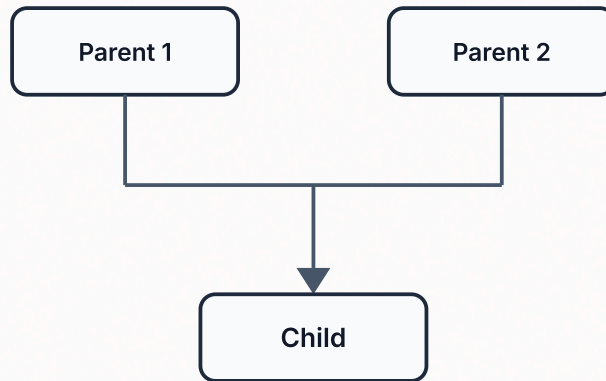
Single Inheritance

Single inheritance occurs when child class inherits only a single parent class.



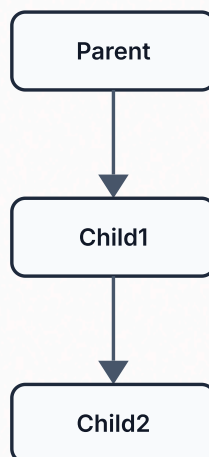
Multiple Inheritance

Multiple Inheritance occurs when the child class inherits from more than one parent classes.



Multilevel Inheritance

When a child class becomes a parent for another child class.



Python Programming Handbook

CHAPTER 11

Beginner Friendly Learning Guide

super() Method

super() method is used to access the methods of a super class in the derived class.

```
super().__init__()

# __init__() Calls constructor of the base class
```

Class Method

A class method is a method which is bound to the class and not the object of the class.

@classmethod decorator is used to create a class method.

Syntax:

```
@classmethod
def(cls, p1, p2):
```

@property Decorators

Consider the following class:

```
class Employee:
    @property
    def name(self):
        return self.ename
```

If e = Employee() is an object of class employee, we can print(e.name) to print the ename by internally calling name() function.

@.getters and @.setters

The method name with '@property' decorator is called getter method.

We can define a function + @ name.setter decorator like below:

```
@name.setter
def name(self, value):
    self.ename = value
```



Python Programming Handbook

CHAPTER 11

Beginner Friendly Learning Guide

Operator Overloading In Python

Operators in Python can be overloaded using dunder methods.

These methods are called when a given operator is used on the objects.

Operators in Python can be overloaded using the following methods:

```
p1+p2 # p1.__add__(p2)
p1-p2 # p1.__sub__(p2)
p1*p2 # p1.__mul__(p2)
p1/p2 # p1.__truediv__(p2)
p1//p2 # p1.__floordiv__(p2)
```

Other dunder/magic methods in Python:

```
__str__() # used to set what gets displayed upon calling str(obj)
```

```
__len__() # used to set what gets displayed upon calling len(obj)
```



Practice Set

1. Create a class (2-D vector) and use it to create another class representing a 3-D vector.
2. Create a class *'Pets'* from a class *'Animals'* and further create a class *'Dog'* from *'Pets'*. Add a method *'bark'* to class *'Dog'*.
3. Create a class *'Employee'* and add salary and increment properties to it. Write a method *'salaryAfterIncrement'* method with a *@property* decorator with a setter which changes the value of increment based on the salary.
4. Write a class *'Complex'* to represent complex numbers, along with overloaded operators *'+'* and *'*'* which adds and multiplies them.
5. Write a class vector representing a vector of n dimensions. Overload the *+* and *** operator which calculates the sum and the dot(.) product of them.
6. Write *__str__()* method to print the vector as follows:

```
7i + 8j + 10k
```

Assume vector of dimension 3 for this problem.

7. Override the *__len__()* method on vector of problem 5 to display the dimension of the vector.



Python Programming Handbook

Beginner Friendly Learning Guide

PROJECT

Project 2 : The Perfect Guess

We are going to write a program that generates a random number and asks the user to guess it.

If the player's guess is higher than the actual number, the program displays **"Lower number please"** .

Similarly, if the user's guess is too low, the program prints **"Higher number please"** .

When the user guesses the correct number, the program displays the number of guesses the player used to arrive at the number.

HINT

```
import random
```



Python Programming Handbook

CHAPTER 12

Beginner Friendly Learning Guide

Advanced Python 1

Following are some of the newly added features in Python programming language.

Walrus Operator

The walrus operator (`:=`), introduced in Python 3.8, allows you to assign values to variables as part of an expression.

```
# Using Walrus operator
if (n := len([1, 2, 3, 4, 5])) > 3:
    print(f"List is too long ({n} elements, expected <= 3)")
```

In this example, `n` is assigned the value of `len([1, 2, 3, 4, 5])` and then used in the comparison within the `if` statement.

Types Definitions In Python

Type hints are added using the colon (`:`) syntax for variables and the `→` syntax for function return types.

```
# Variable type hint
age: int = 25

# Function type hints
def greeting(name: str) -> str:
    return f"Hello, {name}!"

# Usage
print(greeting("Alice"))
```

Advanced Type Hints

Python's typing module provides more advanced type hints, such as List, Tuple, Dict and Union.

```
from typing import List, Tuple, Dict, Union
```

```
from typing import List, Tuple, Dict, Union

# List of integers
numbers: List[int] = [1, 2, 3, 4, 5]

# Tuple of a string and an integer
person: Tuple[str, int] = ("Alice", 30)

# Dictionary with string keys and integer values
scores: Dict[str, int] = {"Alice": 90, "Bob": 85}

# Union type
identifier: Union[int, str] = "ID123"
```



Python Programming Handbook

CHAPTER 12

Beginner Friendly Learning Guide

Match Case

Python 3.10 introduced the match statement, which is similar to the switch statement found in other programming languages.

```
def http_status(status):
    match status:
        case 200:
            return "OK"
        case 404:
            return "Not Found"
        case 500:
            return "Internal Server Error"
        case _:
            return "Unknown status"

print(http_status(200))
print(http_status(404))
```

Dictionary Merge & Update Operators

New operators `|` and `|=` allow for merging and updating dictionaries.

```
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}

merged = dict1 | dict2

print(merged)
```

```
with (
    open('file1.txt') as f1,
    open('file2.txt') as f2
):
    # Process files
    pass
```



Python Programming Handbook

CHAPTER 12

Beginner Friendly Learning Guide

Exception Handling In Python

There are many built-in exceptions which are raised in python when something goes wrong.

```
try:
    # Code which might throw exception
    print(10 / 0)

except Exception as e:
    print(e)
```

```
try:
    # Code

except ZeroDivisionError:
    # Code

except TypeError:
    # Code

except:
    # All other exceptions
```

Raising Exceptions

We can raise custom exceptions using the raise keyword in python.

```
age = -5

if age < 0:
    raise ValueError("Age cannot be negative")
```

Try With Else Clause

```
try:
    # Some code

except:
    # Some code

else:
    # Executed if try was successful
```

Try With Finally

```
try:
    # Some code

except:
    # Some code

finally:
    # Executed regardless of error
```

Python Programming Handbook

CHAPTER 12

Beginner Friendly Learning Guide

If `__name__ == '__main__'` In Python

'`__name__`' evaluates to the name of the module in python from where the program is ran.

If the module is being run directly from the command line, the '`__name__`' is set to string "`__main__`".

Thus, this behaviour is used to check whether the module is run directly or imported to another file.

```
if __name__ == "__main__":  
    print("Running directly")
```

The Global Keyword

'global' keyword is used to modify the variable outside of the current scope.

```
x = 10  
  
def change():  
    global x  
    x = 20
```

Enumerate Function In Python

The 'enumerate' function adds counter to an iterable and returns it.

```
list1 = ["Harry", "Rohan", "Shubham"]  
  
for i, item in enumerate(list1):  
    print(i, item)
```

List Comprehensions

List Comprehension is an elegant way to create lists based on existing lists.

```
list1 = [1,7,12,11,22]  
  
list2 = [item for item in list1 if item > 8]
```



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 12

Practice Set

1. Write a program to open three files *1.txt*, *2.txt* and *3.txt*. If any of these files are not present, a message without exiting the program must be printed prompting the same.
2. Write a program to print third, fifth and seventh element from a list using enumerate function.
3. Write a list comprehension to print a list which contains the multiplication table of a user entered number.
4. Write a program to display a/b where a and b are integers. If $b=0$, display infinite by handling the '*ZeroDivisionError*'.
5. Store the multiplication tables generated in problem 3 in a file named *Tables.txt*.



Python Programming Handbook

CHAPTER 13

Beginner Friendly Learning Guide

Advanced Python 2

Virtual Environment

An environment which is same as the system interpreter but is isolated from the other Python environments on the system.

Installation

To use virtual environments, we write:

```
pip install virtualenv
```

We create a new environment using:

```
virtualenv myprojectenv
```

The next step after creating the virtual environment is to activate it.

We can now use this virtual environment as a separate Python installation.

Pip Freeze Command

'pip freeze' returns all the package installed in a given python environment along with the versions.

```
pip freeze > requirements.txt
```

The above command creates a file named 'requirements.txt' in the same directory containing the output of 'pip freeze'.

We can distribute this file to other users, and they can recreate the same environment using:

```
pip install -r requirements.txt
```

Lambda Functions

Function created using an expression using 'lambda' keyword.

Syntax:

```
lambda arguments: expression
```

Example:

```
square = lambda x:x*x  
  
square(6)  
sum = lambda a,b,c:a+b+c  
sum(1,2,3)
```

Python Programming Handbook

CHAPTER 13

Beginner Friendly Learning Guide

Join Method (Strings)

Creates a string from iterable objects.

```
l = ["apple", "mango", "banana"]  
  
result = ", and, ".join(l)  
  
print(result)
```

The above line will return: "apple,and,mango,and,banana".

Format Method (Strings)

Formats the values inside the string into a desired output.

```
template.format(p1,p2...)
```

Syntax:

```
"{} is a good {}".format("harry", "boy")  
  
"{1} is a good {0}".format("harry", "boy")
```

Map, Filter & Reduce

Map applies a function to all the items in an input_list.

Syntax:

```
map(function, input_list)
```

Filter creates a list of items for which the function returns true.

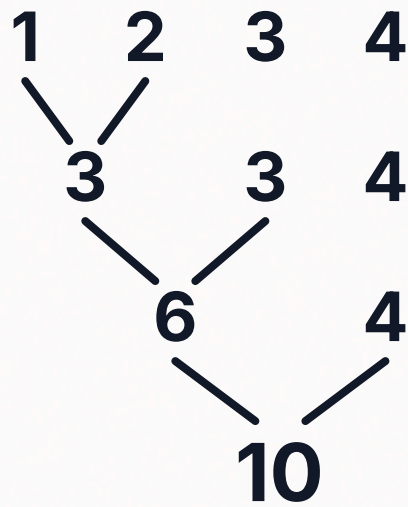
```
list(filter(function))
```

Reduce applies a rolling computation to sequential pair of elements.

```
from functools import reduce  
  
val = reduce(function, list1)
```

If the function computes sum of two numbers and the list is [1,2,3,4]





Sequential Computation



Python Programming Handbook

Beginner Friendly Learning Guide

CHAPTER 13

Practice Set

1. Create two virtual environments, install few packages in the first one. How do you create a similar environment in the second one?
2. Write a program to input name, marks and phone number of a student and format it using the format function like below:

"The name of the student is Harry, his marks are 72 and phone number is 99999888"

3. A list contains the multiplication table of 7. Write a program to convert it to vertical string of same numbers.

7
14
.
.
.

4. Write a program to filter a list of numbers which are divisible by 5.
5. Write a program to find the maximum of the numbers in a list using the reduce function.
6. Run pip freeze for the system interpreter. Take the contents and create a similar virtualenv.
7. Explore the 'Flask' module and create a web server using Flask & Python.



Python Programming Handbook

MEGA PROJECT

Beginner Friendly Learning Guide

Jarvis - Voice Activated Virtual Assistant

Jarvis is a voice-activated virtual assistant designed to perform tasks such as web browsing, playing music, fetching news, and responding to user queries using OpenAI's GPT models.

Features

- **Voice Recognition**
 - Utilizes the speech_recognition library to listen for and recognize voice commands.
 - Activates upon detecting the wake word **"Jarvis"** .
- **Text-to-Speech**
 - Converts text to speech using pyttsx3 for local conversion.
 - Uses gTTS (Google Text-to-Speech) and pygame for playback.
- **Web Browsing**
 - Opens websites like Google, Facebook, YouTube and LinkedIn based on voice commands.
- **Music Playback**
 - Interfaces with a musicLibrary module to play songs via web links.
- **News Fetching**
 - Fetches and reads the latest news headlines using NewsAPI.
- **OpenAI Integration**
 - Handles complex queries and generates responses using OpenAI's GPT models.
 - Acts as a general virtual assistant similar to Alexa or Google Assistant.



Python Programming Handbook

MEGA PROJECT

Beginner Friendly Learning Guide

Workflow

INITIALIZATION

1. Greet the user with "Initializing Jarvis..."
2. Listen for the wake word "Jarvis".
3. Acknowledge activation by saying "Ya."
4. Process commands to determine actions such as opening websites, playing music, fetching news, or generating responses.

SPEECH OUTPUT

1. Provide responses using pyttsx3 or gTTS.
2. Play generated speech using pygame.
3. Continue interaction until terminated by the user.
4. Repeat wake-word detection for the next command cycle.

Libraries Used

speech_recognition

webbrowser

pyttsx3

musicLibrary

requests

openai

gTTS

pygame

os



Python Programming Handbook

MEGA PROJECT

Beginner Friendly Learning Guide

Auto-Reply AI Chatbot

This project automates the process of interacting with a chat application by analyzing chat history and generating humorous AI-powered responses.

The virtual assistant, named Naruto, creates roast-style replies based on the latest conversation history using OpenAI's GPT models.

Features

- **Automated Chat Interaction**
 - Uses pyautogui to perform mouse and keyboard operations automatically.
- **Chat History Analysis**
 - Copies chat history from the application and checks the latest sender.
- **Humorous Response Generation**
 - Integrates with OpenAI's GPT models to generate roast-style responses.
- **Clipboard Operations**
 - Uses pyperclip for copying and pasting generated responses.
 - Detects whether the latest message belongs to a specific target user.
 - Automatically pastes and sends the generated response.



Python Programming Handbook

MEGA PROJECT

Beginner Friendly Learning Guide

Workflow

INITIALIZATION & SETUP

1. Click on the Chrome icon to open the chat application.
2. Wait for a brief period to ensure the application is open and ready for interaction.
3. Periodically select and copy chat history by dragging the mouse over the chat area and using the copy shortcut.
4. Retrieve the copied text from the clipboard.

MESSAGE ANALYSIS

1. Analyze the copied chat history to check if the last message is from a specific user (e.g., "Rohan Das").
2. If the last message is from the target user, send the chat history to OpenAI's GPT-3.5-turbo to generate a humorous response.
3. Copy the generated response to the clipboard.
4. Click on the chat input area and paste the generated response.
5. Press 'Enter' to send the response.
6. Wait briefly before repeating the workflow cycle again.

Libraries Used

pyautogui

Automates mouse and keyboard interactions.

time

Adds delays between automated operations.

pyperclip

Handles clipboard copy and paste operations.

openai

Integrates OpenAI GPT models for responses.

