BDS EDUCATION
Inspire, Innovate, Create

Based on NEP (2020) & NCF (2023)
With Benefits of TIY (Teach it Yourself) Approach

FOUNDATION OF PYTHON

SECONDARY SCHOOL & COLLEGE

Class 9

Assisting Schools Churn out Job/Entrepreneurship Ready Work Force having "Hands-on" Capabilities

Naveen Chandra Gupta

Complimentary Benefits for Schools:

1. Training of your Lab Faculty
2. 7x24 Support to seek clarifications
3. Access to our project library
4. Guidance in student conceived projects
5. Guidance to procure project components & kits
6. Access to monthly faculty update Webinar
7. Option for team participation in competitions

We also help Schools:

1. Convert their existing Computer lab into a "Computer cum Robotic Skill Development" Lab.
2. Provide BDS Trained faculty to be on school payroll

Concept of TIY (Teach it Yourself)

1

- Make teaching content simple
- Base it on what a child seen around him daily
- Project it graphically in colorfulbook form
- Supplement with Home Tinkering & Self Evaluation

2

- Children learn reading given lesson like a story book
- Next day the teacher supplements their learning
- Thereafter, they do projects in buddy teams
- Stronger teams help weaker teams

3

Book includes real life projects to:

- kindle interest in reading and doing
- Develop habit of annotating it with notes
- Preserving it for easy reference

4

Concept applies to TIY books for both, Teachers & Students

Name .....

Class.....Roll No.....

School.....

LIST OF CONTENT

Part 1 Python Basics (Yellow)

Part 2 Understanding Python Tools (Green)

Part 3 Using Python Tools (Purple)

Part-1

CHAPTER

01 Let us Get Started

1 Lesson Overview

This lesson shall cover, What is Python, Downloading of IDLE, Our First few Codes & an introduction to a few key terms to get you started with more ahead.

2 Layout of the Book

The book has been divided into three parts:

Part 1. Python basics. (introduction, cmd prompt, rules and converntions).

Part 2. Understanding python tools (var, conditionals, loops, data types, operators and functions).

Part 3. using python tools (sequences, strings, lists, tuples, dict, sets and errors).

3 What is Python

Python is an **interpreted, object-oriented, high-level programming language,** with **dynamic semantics.** Very confusing. Let us make it simpler.

Look at a **one-line code:**

```
>>> activity = 'We are learning Python from BDS Education'
```

- **Interpreted** means, once the code is written, Python stores & executes its green parts, without first compiling them into machine language.
- Whenever called, it returns the green pars for conversion to machine language. These green parts are stored as Objects. This makes it an **Object-oriented** language.
- Coding is a means of communication between humans who understand English & machines that understand Binary. Closer our code is to English, Higher its Level. In Python we code in English (green parts). This makes it a **High-Level Programming Language.**
- **Dynamic Semantics** is its internal framework of logic & words, that is very English like.

Isn't it now simple.

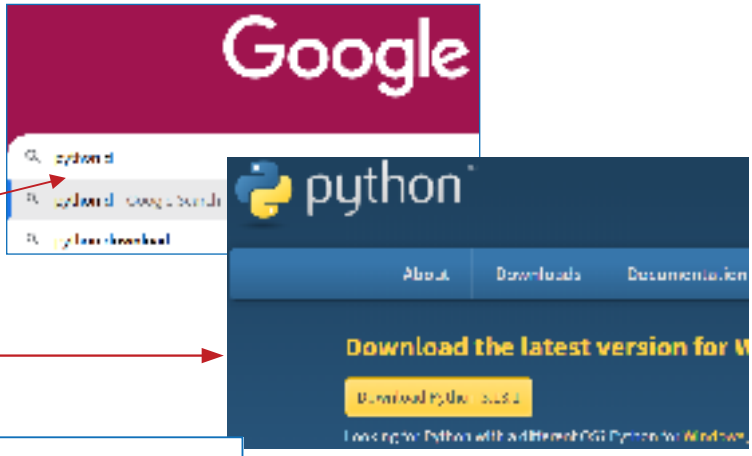
4 Programming Environments

- Python, offers two download options to code:
  - IDLE (Integrated Development & Learning Environment). It offers line by line coding & is designed for learning the use of Python coding tools.
  - IDE (Integrated Development Environment). These are open-source platforms inspired by Python & developed by different companies for professional coding.
- We shall start with an IDLE, & later Migration from IDLE to IDE.

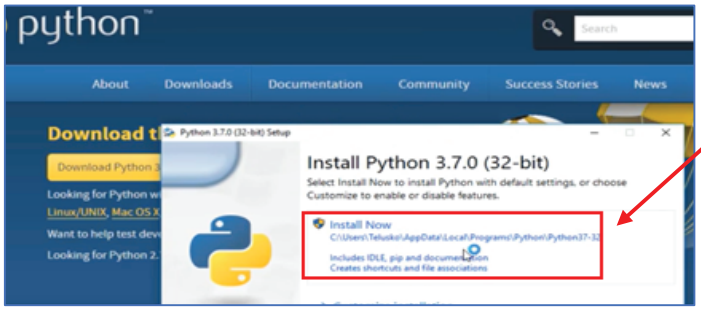
5 Downloading IDLE

To learn Python, the first step is to download Python IDLE.
To do this:

- Google Python download.



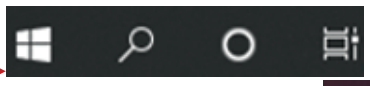
- Click on Download Python.
- Select Download 3.7.0 or the latest version that appears.



Select **install Now.** Installing will start but will take some time.

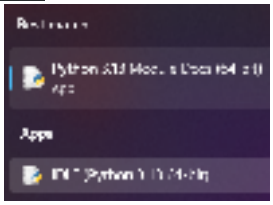
At the end, **Set Up Successful** window appears. Click on **Close.** Download is complete.

To use: Go to bottom left of menu bar. Click search icon.

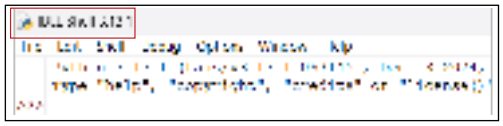


- Type Python against search icon.

Window on the right opens.
In search result, select IDLE.



- Python IDLE Shell appears.



Most important part of the shell is this Symbol >>>
It indicates start of a line. Entire Python coding is done against this. It is called Cmd (command) Prompt.

Note: Shell background is selectable between **white or black.** We have chosen white.

6 Building Blocks Of Python Script

In programming, token is a single element of a programming language.

Example:

```
>>> 'Four is < 6'
```

Here:

Each alphabet (F) each symbol (<) & each number (6) constitutes an individual token.

7 A Word about Python Script & Source Code

Python script is a collection of commands, designed to be executed like a program. In a conventional script:

- Inputs against >>> make cmd Line (line1).
- Statements, actions & output instructions go into subsequent lines called Code lines (Lines 2 & 3).

```
>>> def greet(x):
        x = 'Hello World'
        return x

>>> print(greet(x))
Hello World
```

In laymen words what we write against the cmd prompt, & between them, is called a Script. Compilation of commands & scripts, into executable program is called Source Code.

8 List of Python Symbols

Just like you know all alphabets & nums used in Python, you must know all the symbols used in it. https://docs.python.org/3/genindex-Symbols.html lists around hundred symbols. Some symbols have a single use, but most can be used in multiple ways. It is important to know these different ways. The link provides a good reference. Do read & refer.

9 Our First Few Codes

Code no 1. Let us start with **Numbers (nums).** Say we want Python to **add** two numbers 4 & 5. We have three options:

Option 1. Code directly: Open Python IDLE. In shell type 4 + 5 against >>> → on enter, result 9 & symbol for new line >>> is displayed. This means, code is correctstored & python is ready for the next command.

```
>>> 4+5
9
>>>
```

Option 2. Create two variables (x & y) & code. →

```
>>> x = 4
>>> y = 5
>>> print(x+y)
9
```

Option 3. Create variables & code using third variable (z). →

```
>>> x = 4
>>> y = 5
>>> z = (x+y)
>>> print(z)
9
```

Code No 2. Now let us try Text. Say we want to print 'Hello World'. In the first code, we had entered the nums directly against the cmd prompt. Doing so for text returns error. This is because text needs to be entered in single or double quotes. → Reasons will be learnt later. Try by changing text.

```
>>> Hello World
SyntaxError: invalid syntax
>>> 'Hello World'
'Hello World'
```

Code No 3. Coding **Two Text Lines.** Say we want to enter 'Hello world' along with message 'This is my third Code'. Its code is as shown in the blue box. →

```
>>> greet = 'Hello World'
>>> msg = 'This is my third Code'
>>> code = greet + msg
>>> print(code)
Hello WorldThis is my third Code
```

Note 1. All text inputs automatically appear in green.
Action command print is in magenta. Output in blue & Error in red and the rest in black.

Improving the above Code. The above code has not given any space between world & this. To make this code look neater, In line 1 we have added a colon (:) after Hello & a space after the colon. These are the finer points we shall keep learning as we go ahead. **Now try this code** using the **other two options** mentioned in code no 1.

```
>>> greet = 'Hello World: '
>>> msg = 'This is my third Code'
>>> code = greet + msg
>>> print(code)
Hello World: This is my third Code
```

Code No 4. Coding Text & Nums.
Say we want to code 'Maths 95 Percent'. When we try to code **the way we have learnt** we get an error. However, if we put both nums & text in quotes (line 4), it works fine.
Remember python has rules (next lesson).

```
>>> subject = 'Maths: '
>>> score = 95 'Percent'
SyntaxError: invalid syntax
>>> score = '95 Percent'
>>> result = subject + score
>>> print(result)
Maths: 95 Percent
```

Code No 5. Coding More Lines.
Say we want to code our mark sheet. Code for this is simple. It has used all ingredients explained in ex above. Note space in 92% under CS & others. This is to give space between the subjects in the output because we want it in the output. Python is full of such tricks. Try to pick & follow such concepts.

```
>>> subject1 = 'Maths: 95 %, '
>>> subject2 = 'Engish: 88 %, '
>>> subject3 = 'CS: 92%'
>>> result = subject1+subject2+subject3
>>> print(result)
Maths: 95 %, Engish: 88 %, CS: 92%
```

To consolidate, try above codes with changed parameters.

10 A Word about Python Interpreter & Compilers

Take the ex of script 'Hello World'. In binary format it looks like this & is very un-nerving!

Python Interpreter takes this (Hello World) **high level script** as input; converts it into **low level binary script** for computer to understand give an output, without you having to bother about its compilation or how it is done?

Most other languages use compilers to do this. This takes in the entire program & requires a lot of time to make & analyze the source code. The interpreter takes a single line of code and very little time to make or analyze it. However, compiled codes run faster. & are suited for applications where speed is important. Example java is a compiled code.

Part-1

CHAPTER 02

Command Prompt, Syntax & Comments

1 Lesson Overview

In has lesson we shall examine its five **interlinked** ingredients:

- **Story line.** Statements that summarise what we want to code & its outline plan.
- **Command Prompt.** Where & how to enter the code.
- **Running options.** Internal functioning of our interface with the machine – cmd prompt.
- **Python Syntax.** Machine grammar that ensures code is readable & processable.
- **Comments.** Notes that record thinking process to help other coders understand.

Story is a written or mental declaration of what we want to code, in what sequence, & the result we desire to achieve. Ex make an app for a shopping list reminder. Its storyline will be:

> Shopping app should enable me to make a list containing item name & qty. It should also allow me to recall the list & make a new list after use.

Its simple code is:

```
>>> shoppingList1 = ['Apples', 12, 'Surf', 3, 'Oil', 1]
>>> print(shoppingList1)
['Apples', 12, 'Surf', 3, 'Oil', 1]
>>> shoppinglist2 = {'Milk: 2', 'Butter: 4'}
```

- 1st line captures data defining the **code.**
- 2nd defines the cmd on which it will be **executed** & method for **return** of output.
- 3rd captures data to make a new list after use of the first & so on.

There are no formal rules for writing a story. It is a Tool to streamline thinking process. It is part of Pythons **Good Practices.** It breaks task into smaller, logical & manageable chunks.

2 Understanding Command Prompt

In Python IDLE, code moves one **line at a time.** Each line of code once entered & accepted is **error free.** In this ex, Name in capital & small are two separate names. We defined it with small g (line 1), & were calling in capital G (Line 2). It thus returned an error. Once removed (line 3), code is accepted, & prompt moves ahead for next cmd.

```
>>> greet = 'Hello World'
>>> print(Greet)
Traceback (most recent call last):
 >>> print(greet)
Hello World
>>>
```

3 Running Command Prompt

It means Executing the Code or sending a **return.** Basic ways of return for **display** (print) are:

```
>>> fruit = 'Apple'
>>> print(fruit)
Apple
>>> fruit
'Apple'
```

- print(fruit) or just fruit (first box().
- print(x) or just x (second box).

Note: 1. Commands like print are case sensitive. Use of P in capitals will give error. Try it.

```
>>> x = 22
>>> print(x)
22
>>> x
22
```

At cmd prompt the last value of any input or variable prevails (44). (first box)Once changed, the changed value prevails (66). Previous value is overwritten.

```
>>> x = 44
>>> print(x)
44
```

```
>>> x = 66
>>> print(x)
66
```

4 Options for Running the Command Prompt

We have two options:

- Interactive Interpreter mode.
- Script file mode.

Option 1 - Interactive Interpreter Prompt Mode

It is the preferred mode of use by **Beginners,** for learning the basics line by line. It is also used by **Programmers,** when they are concerned about the output of each line.
This is called interactive because it is executed through interaction with the interpreter:

- Once the object is defined (x = 4+5), the data is saved (1st line).
- Thereafter, the shell waits for action cmd print(X) from the user.

This cmd can be given soon after def the var (as in this case) or at time of its use later.

```
>>> x = (4 + 5)
>>> print(x)
9
```

Interactive mode is best to run single-line statements of a code. Its disadvantages are:

- It is not suitable to write long codes having multiple lines.
- It is not suited for off line coding which is very frequently the case with professionals.

Option 2 - Script File or Script Mode

Using script mode, we can write codes off line. We can write multiple lines of code into a file which can be saved & executed later. For this, we need to open an editor like notepad, create a file, name it & save it with .py extension (py stands for Python). In lesson on migration to IDE, in the next volume we shall be introduced to this.

Advantages of Script Mode. It has a strong prompting mechanism to assist data entry. Its debugging is easy & is the preferred mode for professional coding. Python IDE is based on this mode.

Disadvantages of Script Mode. We have to save the code every time we make a change. This procedure can be tedious when as beginners we need to run single or few lines of code.

5 Understanding Python Syntax

When we run a program, it is first read by a **parser** (an element of a program, usually part of interpreter that checks validity of inputs). The parser will pass the Python code only if it is **hundred percent** according to **pre-defined rules,** otherwise it will give errors. This imparts python with the **strongest error correction mechanism.**

Common Syntax Errors made by Beginners

- **Sensitivity to use of quotes.** When def object (fruit), all alphabetic characters must be in quotes.
- **Defining numeric characters** does nor require quotes. Though use of quotes will not result in an error.
- **Mixed Quotes.** Using single quote at one end & double at the other will result in error.
- **When error occurs** read text in red. Here it has guided us that parentheses are missing.

```
>>> fruit = apple
Traceback (most recent call last):
```

```
>>> fruit = 'apple'
>>> x = 44
```

```
>>> objectl = 'We have 44 apples"
SyntaxError: EOL while scanning stri
>>> objectl = 'We have 44 apples'
```

```
>>> print fruits
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(fruits)?
```

6 Making Interactive Codes

The code that runs our ATM is an interactive code. It enables actual **Users to Input Data.** Example code to enter password.

The code stores it as an **input() function**. When the code is finally run, interpreter reaches the **input()** function, & gives a request for that input. It then waits for the answer from the user. It moves ahead **only after** that input has been received from user & is accepted by it.

7 Understanding Comments

Comments are unique to Python & an important coding tool. They help:

- Us & others to **understand the code** we have written.
- By looking at comment, we can understand the **coders mind** & the **intention** of each line.
- They help **trace errors** & fix them.
- They help **Re-use** the code in other applications.
- They are a great tool for use by learners to get used to, & to share codes with peers.
- Comments should **be short, sweet, & to the point.** Python Enhancement Proposal (PEP 8) advises that code be kept at 79 characters or less per line. It also suggests a max of 72 characters for inline comments & docstrings, & spread them over multiple lines if exceeding.

8 Types of Comments

Python supports two main types of comments:

- Single line Comments. It is called single line because we cannot press enter, in-between the process of writing it. If we press enter, new cmd prompt appears.

```
>>> text = "hallo World" # "Hello World" was first used in 1972. It has since be
come a tradition as the first program students write.
```

- Multi-line Comment. It has no support for multi-line comments. Use # for each line.

```
>>> # Line 1 of multiline comment.
>>> # line 2 of multiplne comment
```

9 Writing of Comments

Firstly, comments can be written against the cmd prompt (1st line), or against the object (2nd line). Secondly, there are two methods of writing comments:

```
>>> # Comment against Cmd Prompt.
>>> text = 'Hallo World' # Comment against object.
>>> print(text)
Hallo World
```

- Using Hash. Called Hash comments they appear in red (as shown above).
- Using triple quotes.
- Comments are ignored when the code is run.

Comments Using Triple Quotes below
These appear in green & are given against cmd prompt only. Giving against object returns error. Comments could be:

- Given using three single quotes.

```
>>> ''' Comment using three sets of single quotes againd cmd propmt'''
```

- Given using three double quotes.

```
>>> """ Ex of a single line comment using three double quotes."""
```

10 Pythons Power of One-liners & Underscore.

- Python Supports Multiple Assignments & Multiple Returns.

This enables us to write one-line codes called one liners.
As learners, try & follow the logic of making one liners.
As programmers, every code line saved is important.

- Python has the Memory Power of Underscore.

It supports underscore (_).
This remembers last result.
It can be used where you forget it.
To recall follow:

```
>>> c = _
>>> c
40
```

Its example is as attached:

```
>>> a = 20
>>> b = 20
>>> a + b
40
```

Part-1

CHAPTER 03

Rules & Conventions

1 Lesson Overview

In has lesson we shall understand the conventions & rules for use of Colour, Names, Parentheses,White Space & Indentation.

2 Python Rules & Conventions

Machines only have a binary logic. **A logic of yes or no, true or false, go or no go, zero or one.** Instructions given to them have to be **PERFECT.** This is possible if stringent rules & conventions are laid down & followed.

Python rules & conventions include conventions for comments, naming, code layout, use of whitespaces, indentations & separations. They can broadly be divided into two types:

- Rules & conventions for style.
- Rules & conventions for naming.

We have covered the rules for comments. Let us now see the others.

PEP (Python Enhancement Proposal) has several proposals written in 2001 by Guido van Rossum, Barry Warsaw, & Nick Coghlan. PEP 8, is a document that provides guidelines & best practices on writing Python code. These are contained in a chapter – Style guide for python code. https://www.python.org/dev/peps/pep-0008/

3 Colour Conventions

As a gen rule, Python objects like names, nums, operators etc are in black. Return commands in purple. String inputs in green. Outputs & function names in blue.

```
>>> My_list = ['Apples', 10, 'Oranges',12]
>>> print(My_list)
['Apples', 10, 'Oranges', 12]
```

Func def, loop statements, logic elements etc in light red.

```
>>> def myfunction():
```

Comments in red or green. These give one look access to programmers to different segments, help debugging & reduce input entry errors.

4 Naming Conventions

While coding, we have to name things like Variables (Var), Function (func) etc. A sensible name tells what a var, func or class represents. They reduce errors & help debugging. Recommended Conventions are:

Type	Convention	Example
Functions	Use lower case words. Separate using underscore	my_func, greet
Variables	Use lower case letter, words or words. Separate using underscore	x, fruits, my_var
Constants	Use upper case letter, word or words. Separate using underscore	Y, CONSTANT
Class	Start word or words with capital letter. No underscore. Camel style.	MyVehicles, Phase1
Methods	Use lowercase word or words. Separate using underscore	my_methods
Modules	Use short word or words in lower case. Separate using underscore	module.py
Package	Use short word or words in lower case. No underscore	mypack

4.1. Naming Var – What is Permitted?

- **Rule-1:** Var name with alphabet or underscore (_) character.
- **Rule-2:** Var name can contain A to Z, a to z, 0 to 9 & underscore (_).
- **Rule-3:** Var names are case sensitive. Ex – My & my, are different var.

```
>>> x = 44
>>> School = 'Bal Bharati'
>>> mySchool = 'Bal Bharati'
>>> my_school = 'Bal Bharati'
>>> _My_School = 'Bal Bharati'
```

4.2. What is Not Permitted?

- **Rule-1:** You cannot start var name with a num or - .
- **Rule-2:** You cannot use special ch such as $,%,#,&,@ etc.
- **Rule-3:** You cannot use a keyword as a var name. In ex 4, global is a keyword.

```
>>> 8_Branch = 'Sarojni Nagar'
SyntaxError: invalid decimal literal
>>> -school = 'Bal Bharti'
SyntaxError: cannot assign to operator
>>> branch@ = 'Sarojni Nagar'
SyntaxError: invalid syntax
>>> global = 'Sectoe 4, Noida'
SyntaxError: invalid syntax
```

5 Use of brackets

These are of three types. They are reserved for:

- Round brackets () for callable functions & classes or for creating tuples.
- Square brackets [] for requesting individual items (ex index value) & creating lists.
- Curley brackets {} for creating dictionaries & sets.

6 White Spaces

Whitespace is a method of **creating spaces around characters** & lines in code using space, tab & CRLF (carriage returns & line feeds). These allow us to **format our code** in a way that **makes them readable** in a single glance. In most cases **Python provides for default** white spacing. In case like assigning values, these have to be provided by us.

6.1. Whitespace Around Operators.

- Provide a **single space** around assignment operators, comparison operators, boolean operators.

```
>>> x = 22
```

```
>>> print(x == y)
```

```
>>> x>4 and y>4
```

- When more than one operator is involved, add space only after operator **with lowest priority.**

```
>>> y = x**2 + 22
>>> z = (x+y) - (x-y)
```

- To express **if** statements in logical segments.
- In slicing, **colons** act as a **binary operator.** Therefore there should be same whitespace on either side.

```
>>> if z>22 and x%4==0:
```

```
>>> list[22 : 33]
>>> list[x+1 : x+2 : x+3]
```

6.2. **White Spaces around punctuations.** Follow typographic rules More then conventions, use logic. Code must have a **clean, clear & uncongested** look.

```
>>> x = [2, 3]
```

6.3. **Mixing Tab & Space Key.** As a good convention, when coding in Python you can **either use tab or space.** Do not mix. PEP8 prefers space.

7 Indentation

It refers to **adding white spaces** before a statement. Without indentation, Python does not know which statement to execute next or which statement belongs to which block, leading to Indentation Error. With four spaces left as indent before print cmd, Python knows 2nd line is going to be a statement it must execute.

```
>>> def func(Greetings):
        print('Welcome to India')
```

7.1. Indentation Rules:

- Leaving a space before **cmd prompt** results in error.

```
>>>  x = 22 # Example 1

SyntaxError: unexpected indent
```

- Not leaving space in a **code line** results in error.

```
>>> if 44> 22:
Print ('Proceed')
SyntaxError: expected an i
>>>
```

```
>>> if 44> 22:
        print('Proceed')


Proceed
>>> # No error
```

- Where required, Python leaves **four spaces by default** (box 3, 2nd line, before print).

7.2. Schematic Diagram to Understand Line structure & Indentations

Real life Code

```
>>> x = 25
>>> y = 18
>>> if y>x:
        print('Temp Greater')
else:
        print('Temp Lower')
```

Its Schematic

```
Statement
if condition
    if condition:
        Statement
    else:
        Statement
Statement
```

Code how Interpreter perceives

```
Code block1 starts
Code block1 continues
    Code block2 starts
        Code block 3 starts
    Code block2 continues
        Code block3 continues
Code block1 continues
```

Note: For indent, you decide num of spaces. One is min. Python leaves four spaces by default. An important diagram to understand.

Ten Commandments of Python

1 It is the Fastest Growing Language.

With advancements in technologies like artificial intelligence, machine learning, and others, demand for Python will be ever-increasing.

2 It uses English language

Python code is more similar to English than others, making it easy (second only to Scratch) to learn & use. It appears as though you are writing in English.

3 It is Beginner Friendly.

It has an English readable syntax that is easy to write, understand and implement.

4 It does not use a Compiler.

Python interpreter reads and converts the code into machine readable & executable file in real-time. This gives that unique coding in English feel. It simplifies writing, testing & debugging of code.

5 It has the Largest Community Support.

Having a problem writing – Just google it with good key words. You will get the answer.

6 It has Extensive Library & Module Support.

This gives access to domains, such as DataScience, Machine learning, Artificial Intelligence, web development, and more. For example, NumPy and Pandas benefit from numerical computing and data analysis. In addition are libraries like Django, TensorFlow & Scikit Learn that help in web development and ML. Learn one, learning others, becomes simpler.

7 It is a Free, Open-Source Language.

It can be used on any application for free for commercial or personal use. Developers can easily modify it and distribute it for free.

8 It consists of C Java Extended Version.

The derivatives for the C language are known as CPython. These can be used in interpreter as well as compiler mode. Some of the versions of Python are.

- ➢ Jython (Java Language.)
- ➢ Brython (For Browsers).
- ➢ RubyPython (Ruby Programming Language).
- ➢ PyPy (Self implementation).
- ➢ IronPython (C# programming language).

9 It Is Portable.

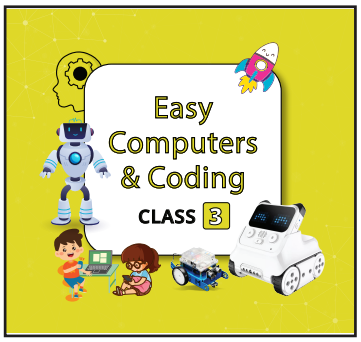
This implies that programmers can reuse their code on different platforms without any re-write.

10 It is the future.

To Read More
Contact - 88267 30055

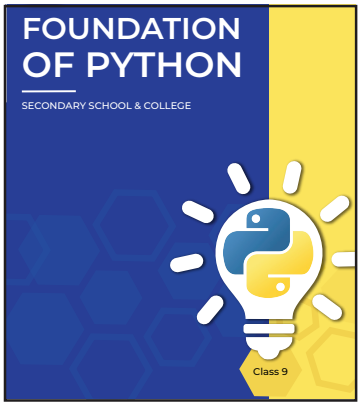
OUR BOOK TITLES

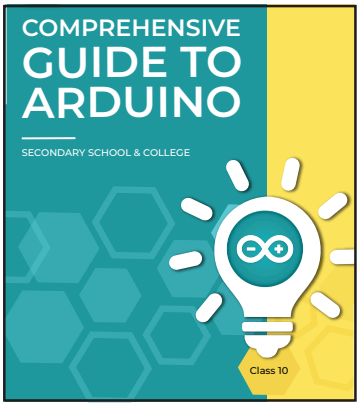
Level 1 - Lower Primary

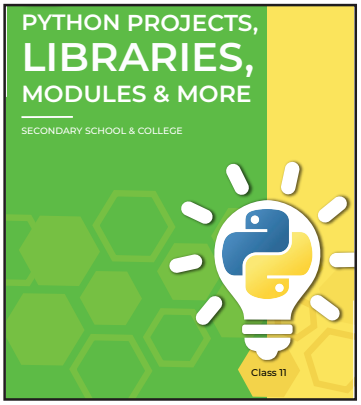


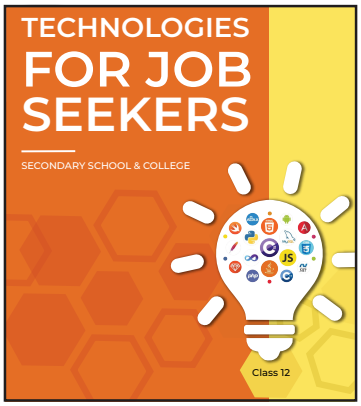
Level 2 - Upper Primary

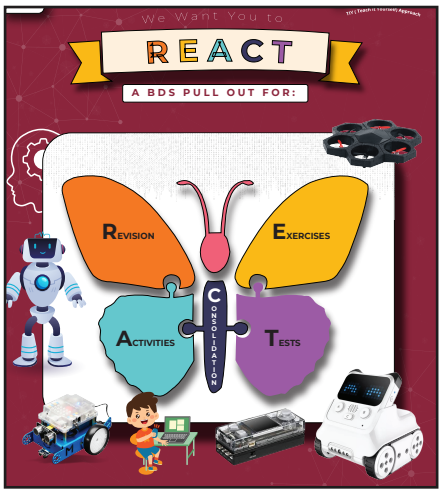
Level 3 - Senior, Senior Secondary & College











Our Vision
Code Karega India Badega

01
Books Have 3 Parts:
- Computers (Yellow)
- Coding (Green)
- Technologies (Purple)

02
Lesson on Home Tinkering Lab

03
Imparts Project Based Learning

BDS CONNECT

04
Q&A cum Test Supplement React with all Books

05
Dedicated Mail ID for Seeking Clarifications

06
Weekly on line "BDS Quiz cum Do You Know"

07
Option for National & International Competitions

Core Team of BDS Education, in 2003 -04, Designed & Executed the Connectivity for Project Akshaya aimed at "Making one Member of Every Family **Computer Literate".** In 2006, it won the Government of Kerala the "Golden Nica" award (Oscar of IT). The team is back to make one member of every Indian family **Coding Literate.**

BDS EDUCATION
Inspire, Innovate, Create

Self Publication of BDS Education
www.bdseducation.in
Education Division of Bharat Drone Systems Pvt Ltd

Content created by
BDS Education Research Foundation
(an ex-Servicemen enterprise)

₹ 520/-

Follow us on : https://bit.ly/4fGsNEc https://bit.ly/4fwQKh4 https://bit.ly/3CicxKW https://bit.ly/4hHluxx