① Introduction & History

1) General purpose High Level Prog Lang.
   - Web App, AI, Data Science App, Stand alone app,
   - Scripting ... etc.

2) Very Very programmer friendly !!

3) Father of Python → Guido Van Rossum in 1989
   (Research Institute, Netherlands)

   Java older / Ver

   Java: 1995

   Python: 1991 → so python is older than Java.

   Feb 20th → was made officially available to public in 1991 by Guido

4) Earlier ... first learn C → later other prg.
   Now .... First 'Python' ✓
   → Easiest Prg Language.

5) 100 lines of code in other prg lang = 10 lines of code in python
   → very concise !!

   - Java → LKG
   - C → Nursery
   - Python → Play School

   Ex: print "Hello world" in Java Vs Python ✓ Vs C

→ Shreekanta Datta

6) >> print('Hello World')
>> 10+20

→ 111 by concepts introduced later in Java shell

7) for i in range(10):
    print('Hello')

8) x=30 if 10>20 else 40
print(x)

} very close to Eng language

9) Java: Web App & Enterprise Applications
Testing (selenium)

Python: Scripting Lang
Web App
Testing Automation
Data Science / ML / AI

Embedded: C, C++

Mobile: Android / Ios

10) Why the word 'PYTHON'?

→ Name influenced by Guido (founder)
↳ He was inspired by the then popular comedy show "The Complete Monty Python's Circus" broadcasted in BBC (1969-1974)

11) Procedural lang
Object oriented prog lang
Functional prog

C → Functional
Java → OOPs
Python → All of them

12)
```
def f1():
    print('Datta')
```

```
>> f1()
f1()
f1()
```

```
Class Test:
    def m1(self):
        print('From m-test')

t = Test()
t.m1()
```

13) Python borrows....

1. Functional prg features from C
2. OOP features from c++
3. Scripting lang from Perl & Shell
4. Modular prg lang from Modula-3.

14) Core Python → Language Fundamentals
Operators
Input & output statements
Flow control
String
List
Set
Tuple
Dictionary
Functions
Modules
Packages.

* Introduction / Trailer / Structure / High level over view / Idea of programming constructs / what makes a python program / Std way of learning a Prg language

→ It's a JOY to prg in python (until :-) u start prg bigger things) !!

→ Idea is to understand the internals of the language

→ Build a python application - end to end.

* What is a Python ?

Ex: Friend... Come lets go... Questions... where? High level understanding.

→ Monte Python → A comedy/drama group.

→ Guido → still the head of the Python org.

Python 1, 2, ... 2.8, 3., 3.X (3.7) Not necessarily backword Compatible

→ Earlier famous with Mathematicians & Researchers

→ Strengths of Python - very much English like (extremely verbose) honest, simple & straight

→ High Level Lang, Dynamically Typed Prg lang (Data type allocated/decided at Run Time)

(box)

* Variable - Named memory location where u can store & retrieve data.

- All boxes cannot hold all type of data. (Data type)

→ Statically Typed → A variable is assigned a DataType & it will remain the same throughout its life time

→ E Analogy → Human being born with a Gender

Now.... Gender changing .... Dynamic Typing !!

→ usu all dynamically typed lang are also " Interpreted "

→ No compilation
→ Every thing at run time (line by line conversion.... TO machine level code)

Adv
- portable

Dis Adv
- slow (Needs to be interpreted every time, even if the prg is same)

* Platform → OS + Hardware

* Python → Portable (with a catch)

If Python calls certain OS services particular to a specific OS, then it is not portable.

* Java populorised the concept of 'portability' (USP / brand language) (Platform independance)

Ex: Americans selling Yoga in different flavours

* Multiparadigm Prg Lang → Scripting,
Structured prg,
OOPs,
Aspect oriented Programming
Functional Programming
like Serverless (architectures)
server comes alive/runs only when needed.

* Python → Open Source

Ex: Oracle buying Sun / Java

* Extensible → You can add extensions to the prg lang itself

Ex: Attach C/C++ to the hooks that Python provides. That now gets built into the language itself.

* Embedded: Python can be embedded in other packages.

* Crown Jewel
USP, Hall mark of Python → Rich Standard library !!

* Maintainability & Enhancement of applications are very easy !!

* Summary !!

PYTHON → General purpose prg lang
→ High Level prg lang, Verbose (English like
→ Dynamically Typed
→ Interpreted
→ Portable
→ Multiparadigm
→ Open Source
→ Extensible
→ Embedded
→ Powerful 'Std Lib'
→ Interactive Programming Language !!

- Jump-In Analogy - swiming training
  In the deep end first.
- Test Driven Development (TDD)
- Friction / Dependancies in other long v/s Python
  ∵ of interactive prg (TDD)
- python.org → install
  → 3.7 → Folder → Python 3
  Add to 'path'
  → Pycharm (IDE)
  ↳ builtin IDE
  called → IDLE
  → Build First.py
  → Run it (No 'main' req)

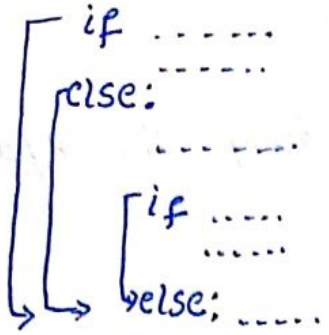
Simple is better than Complex
Explicit is better than Implicit

- Exercise Prg → Look at the time, see the secs,
  if odd → odd time
  even → Even time.

```
from datetime import datetime

sec = datetime.today().second
odds = [1, 3, 5 ..... , 59]
if sec in odds
    print('This is an odd moment')
else:
    print('This is not an moment')
```

- Representing blocks in python is thru 'Indentation'
  - No flower braces in Python



- Making best practices are made a 'Rule'
  - Benifits of following best practices / Rules

- import - modules or fuctions
  - ↳ can have 'submodules'

from datetime import datetime
↓ ↓
Module Submodule

- Python comes with batteries included !!

- >> print ('bond' + str(25))

- // Develop the mindset of TDD in Python //
  - ↓
  - // Breaking the code in small chunks & test it interactive first //

* >> import random
>> dir(random)
>> help(random.randint)

// From not req if you are not importing a submodule from the module

⇒ calling the method
modulename.method

* for i in [1, 2, 3, 4, 5]:
    print(i)

* for ch in 'WowThisIsCrazy':
    print(ch)

* for i in range(5):
    print(i)

range(5)
↳ evaluates range(0, 5)
= [0, 1, ... 4]

{ import time
  import random

* for i in range(5):
    if/sec randomval.randint(0, i)
    time.sleep(randomval)
    sec = datetime.today().second
    if sec in odds
        print('odd')
    else:
        print('not odd')