Python for beginners

- Reference books
    1. Intro to Python by Deitel
    2. Let us Python by Kanitkar
- OS - Windows
- Python Version 3.11 (Any Python version 3.x is fine)
- Cheatsheets 1 Keywords, 2 Data structures, 3 Complex data, 4 Classes, 5 Functions

Rules

- This is not a computer science course
- There are alternative methods of solving the same problems. Feel free to explore at your own time

Jupyter notebooks

Jupyter is a platform for creating and sharing computational documents. Text and code can be handled in one single file. You could try Jupyterlab if you want to try more features.

Install Jupyter using `pip install jupyter`

Start Jupyter using `jupyter notebook`

> File Extension - `.ipynb`

Markdown

Markdown is a markup! language for formatting text using a text editor. Its popular in blogs, readme, documentation etc. It is a text to html converter. Try the cheatsheet

Moore's Law

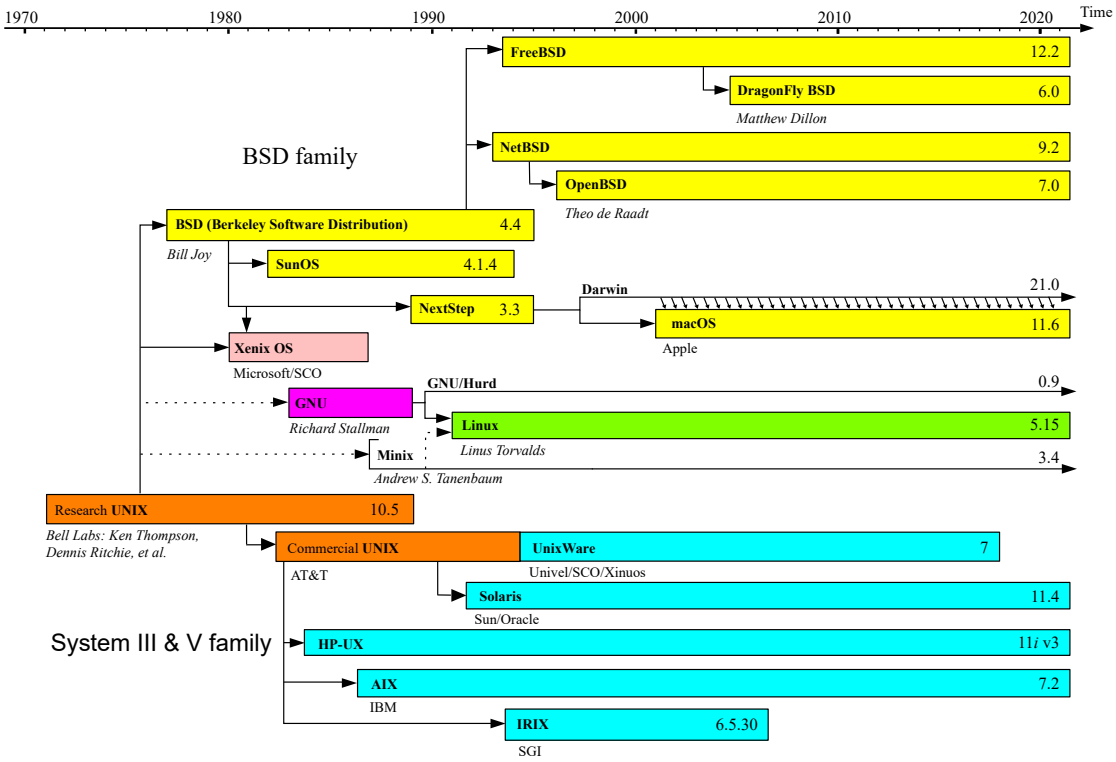
It is the observation that the number of transistors in a dense integrated circuit (IC) doubles about every two years

> Gordon Moore - Co-Founder (Intel)

Operating systems

1. Windows - From earlier DOS
2. Unix - Bell Labs, AT&T, Sun/Oracle Solaris, HP-UX
3. GNU Linux
4. BSD
5. MacOS
6. iOS / ipadOS / WatchOS

7. Android - Using Linux Kernal



> A kernal allocates machine resources
>
> Popular linux distrbutions are Ubuntu, Open Suse, Redhat, Fedora etc

Processors

- Intel
- AMD
- Apple M
- Snapdragon
- Mediatek
- Samsung Exynos

Architectures

- ARM Mobile chips, Now into other applications also - Apple M2, AWS Graviton
- X86 Popularly known as 32 bit (Remember Intel 8086)
- X64 Popularly known as 64 bit (Actually X86-64)

> ASIC - Application Specific Integrated Chip | Eg: - Graphics Card, Crypto mining

Platforms

Platform is a combination of Processors and OS. Applications are usually written for a platform.

Java was popular for being platform independent (in addition to other reasons). This means the same code base will work across all platforms. Java achieved this using **JVM** (*Java Virtual Machines*) sitting over the OS. Python has the same feature

Languages

1. Machine Languages
2. Assembly languages

```
load basicpay
add da
add hra
add allowances
store grosspay
```

3. High level languages - Eg: - C, C++, Java, Python, Ruby, Go, Carbon, Swift, Kotlin

```
grosspay = basicpay + da + hra + allowances
```

Compilers vs Interpreters

C is a compiled language. Compiled program runs faster - better performance

`gcc hello.c`

`./a.out`

> a.out stands for assembler output

Python is interpreted

```
a = 5 + 7
```

> interactive mode

`hello.py`

> script mode

Python

- Created by Guido van Rossum in 90s
- Open Source
- easy to learn
- Used for web applications (Flask / Django)
- popular with AI / ML / Financial community

Python on Windows

Python can be run in 2 different ways

1. Interactive method - Execute one statement at a time
2. Scripting method - run a `.py` file

There are a few ways to run Python on windows

1. IDLE
2. Start python from windows command line by typing `python` in the cmd
3. A web tool Jupyter notebook
4. A text editir like VS Code - for scripting method

Python as a calculator (interactive)

In [ ]:
```
10 + 15
```

Out[ ]:
```
25
```

In [ ]:
```
12.7 - 4
```

Out[ ]:
```
8.7
```

In [ ]:
```
12.7/2
```

Out[ ]:
```
6.35
```

In [ ]:
```
5 * ( 3.6 + 1.11)
```

Out[ ]:
```
23.55
```

In [ ]:
```
print ("Hello")
```

```
Hello
```

In [ ]:
```
a = 5
print(a)
```

```
5
```

In [ ]:
```
5**2 # exponent
```

Out[ ]:
```
25
```

In [ ]:
```
10//3 # Floor division
```

Out[ ]:
```
3
```

In [ ]:
```
10%3 # modulo
```

Out[ ]:
```
1
```

In [ ]:
```
a = 5
a *=3 # same as a = a * 3
print (a)
```

```
15
```

Python libraries

Avoids reinventing the wheel. Remember `#include "math.h"` in C language?

Standard libraries

- os
- datetime
- math
- json
- sqlite3
- string

Data Science libraries

- NumPy
- SciPy
- Pandas
- Matplotlib
- TensorFlow
- NLTK

In [ ]:
```
# Use OS library to get system info
import os
info = os.environ
print ("Operating system :", info['OS'])
print ("CPU Cores :", info['NUMBER_OF_PROCESSORS'])
print ("User :", info['USERNAME'])
```

```
Operating system : Windows_NT
CPU Cores : 8
User : Subin Abid
```

Objects

Classes

Reusable software components

```
class Project:
    name = Talcher
    zeroDate = 27/09/2022
    contractor  = BHEL
    duration =  44
}
```

Methods

Methods perform tasks on classes

```
class Project:
    zeroDate = 27/09/2022
```

```
    contractor  = BHEL
    duration =  48
    delay = 0

    def endDate(self):
        return (self.zeroDate + self.duration + self.delay)
}
```

Instance & reuse

Instance is when we use a class

```
projectTalcher = Project()
print (projectTalcher.zeroDate)
print (projectTalcher.endDate())

ProjectTalcher.delay = -4
print (projectTalcher.endDate())

ProjectTelangana = Project()
ProjectTelangana.delay = 6
print (projectTelangana.endDate())
```

Inheritance

A station is interested in the project history

```
class Station(Project):
    codDate = xxx
```

In [ ]:

```
# Python Code for above problem
import datetime  #datetime is a standard library

class Project:                              # define an object called Project

    duration = 48                           # Duration of a project defined as 48 mo

    def __init__(self, name, zerodate):     # initiate a project with name and zeroc
        self.name = name
        self.zerodate = zerodate
        self.delay = 0

    def endDate(self):
        return self.zerodate + datetime.timedelta(days = self.duration * 30 + self.

# Initiate 2 projects, Talcher and Telangana
projectTalcher   = Project("Talcher",   datetime.datetime(2022,9,27))
projectTelangana = Project("Telangana", datetime.datetime(2018,9,27))

print("Project Talcher initiated ")
print("Name:", projectTalcher.name)
print("Zero Date:", projectTalcher.zerodate)
print("Delay:", projectTalcher.delay)
print("End Date:", projectTalcher.endDate())
print("\n")
print("Project Telangana initiated ")
print("Name:", projectTelangana.name)
print("Zero Date:", projectTelangana.zerodate)
```

```
print("Delay:", projectTelangana.delay)
print("End Date:", projectTelangana.endDate())

# Update Delay on both Projects
projectTalcher.delay = -4
projectTelangana.delay = 6
print("\n" + "Delays added to projects. New End dates are:")
print("Talcher - Delay:" + str(projectTalcher.delay) + "Months. New End date:" + st
print("Telangana - Delay:" + str(projectTelangana.delay) + "Months. New End date:"
```

```
Project Talcher initiated
Name: Talcher
Zero Date: 2022-09-27 00:00:00
Delay: 0
End Date: 2026-09-06 00:00:00


Project Telangana initiated
Name: Telangana
Zero Date: 2018-09-27 00:00:00
Delay: 0
End Date: 2022-09-06 00:00:00

Delays added to projects. New End dates are:
Talcher - Delay:-4Months. New End date:2026-05-09 00:00:00
Telangana - Delay:6Months. New End date:2023-03-05 00:00:00
```

Lesson 2 plan

- Introduction to Python programming
- Git / Github

> By Subin Abid, Project Manager, NTPC Ltd.
