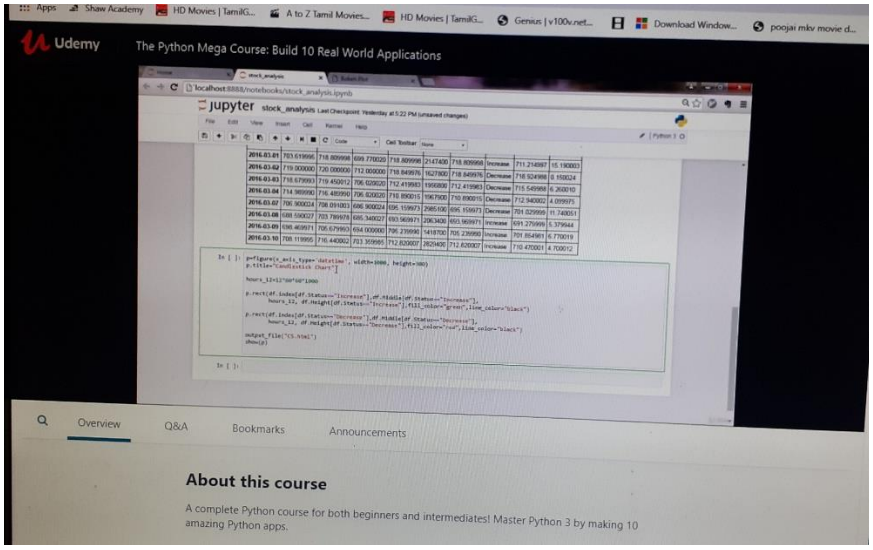
DAILY ASSESSMENT

Date:	**4-6-2020**	**Name:**	**Kavyashree m**
Course:	**Python programming**	**USN:**	**4al15ec036**
Topic:	**Application 9: Build a Web-based Financial Graph**	**Semester & Section:**	**8th A**
Github Repository:	**kavya**		

FORENOON SESSION DETAILS
Image of session


Udemy
The Python Mega Course: Build 10 Real World Applications
localhost:8888/notebooks/stock_analysis.ipynb
jupyter stock_analysis Last Checkpoint: Yesterday at 5:22 PM (unsaved changes)
Overview
Q&A
Bookmarks
Announcements

2016-03-01	703.619995	718.809998	699.770020	718.809998	2147400	718.809998	Increase	711.214997	15.190003
2016-03-02	719.000000	720.000000	712.000000	718.849976	1627800	718.849976	Decrease	718.924988	0.150024
2016-03-03	718.679993	719.450012	706.020020	712.419983	1956800	712.419983	Decrease	715.549988	6.260010
2016-03-04	714.989990	716.489990	706.020020	710.890015	1967900	710.890015	Decrease	712.940002	4.099975
2016-03-07	706.900024	708.091003	686.900024	695.159973	2985100	695.159973	Decrease	701.029999	11.740051
2016-03-08	688.590027	703.789978	685.340027	693.969971	2063400	693.969971	Increase	691.279999	5.379944
2016-03-09	698.469971	705.679993	694.000000	705.239990	1418700	705.239990	Increase	701.854981	6.770019
2016-03-10	708.119995	716.440002	703.359985	712.820007	2829400	712.820007	Increase	710.470001	4.700012

```
In [ ]: p=figure(x_axis_type='datetime', width=1000, height=300)
        p.title="Candlestick Chart"

        hours_12=12*60*60*1000

        p.rect(df.index[df.Status=="Increase"],df.Middle[df.Status=="Increase"],
               hours_12, df.Height[df.Status=="Increase"],fill_color="green",line_color="black")

        p.rect(df.index[df.Status=="Decrease"],df.Middle[df.Status=="Decrease"],
               hours_12, df.Height[df.Status=="Decrease"],fill_color="red",line_color="black")

        output_file("CS.html")
        show(p)

In [ ]:
```

About this course

A complete Python course for both beginners and intermediates! Master Python 3 by making 10 amazing Python apps.

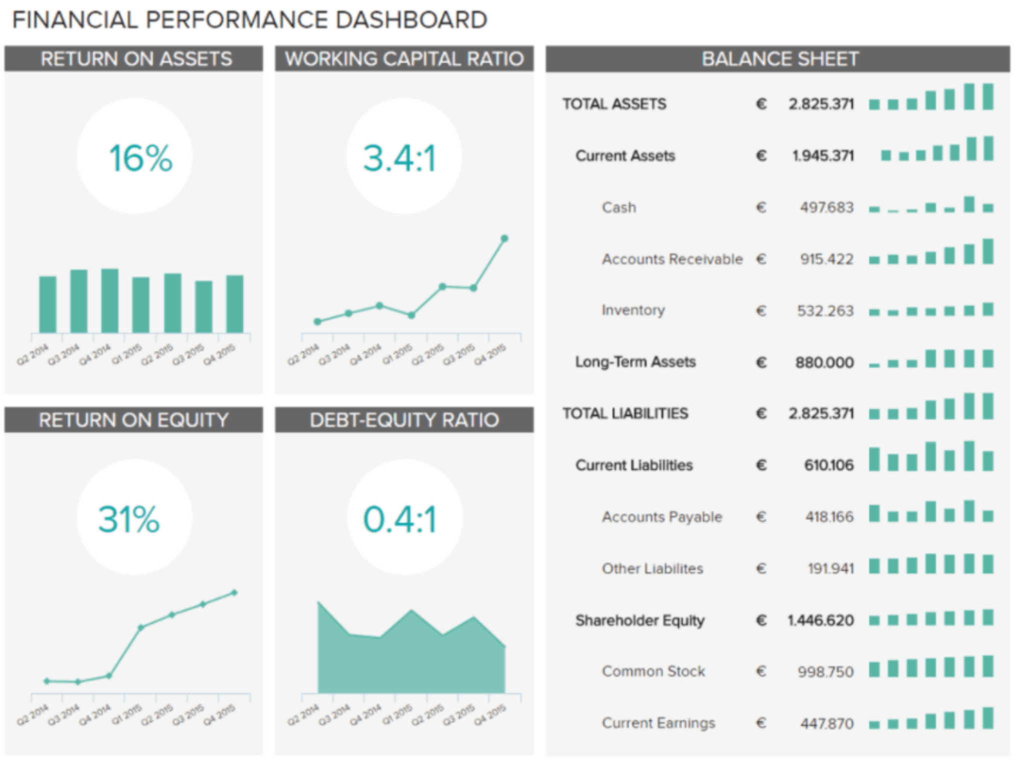
Fig 1: Build a Web-based Financial Graph

Build a Web-based Financial Graph

As humans, we respond to, and process visual data better than anything else. That said, when it comes to digesting and taking action upon vital financial metrics and insights, well-designed finance graphs and charts offer the best solution. And according to Illinois State University, when it comes to visual aids of this kind, three standards apply: graphs and charts should display unambiguous information, meaningful data, and present said insights in the most efficient way possible.

Fundamentally, you need financial graphs as:

- You will be able to track your liquidity, cash flow, budgets, and expenses accurately with ease, visually, and automate processes that were oftentimes done manually and with higher risks of errors.
- By setting the right financial KPIs for your business, you will be able to set valuable financial goals that result in growth and success. While there are numerous charts out there, we will explain the invaluable ones for any business.
- You will be able to make sense of all the financial data and metrics as they will be split into actionable categories and presented in an intuitive, scannable fashion, no matter the metric you need to include and analyze.
- Pen and paper or static data will no longer cut it in today’s fast-paced, competitive and data-rich commercial landscape. As mentioned, manual work is prone to mistakes that you can easily avoid by using self-service analytics software.



TOTAL ASSETS	€	2.825.371
Current Assets	€	1.945.371
Cash	€	497.683
Accounts Receivable	€	915.422
Inventory	€	532.263
Long-Term Assets	€	880.000
TOTAL LIABILITIES	€	2.825.371
Current Liabilities	€	610.106
Accounts Payable	€	418.166
Other Liabilites	€	191.941
Shareholder Equity	€	1.446.620
Common Stock	€	998.750
Current Earnings	€	447.870

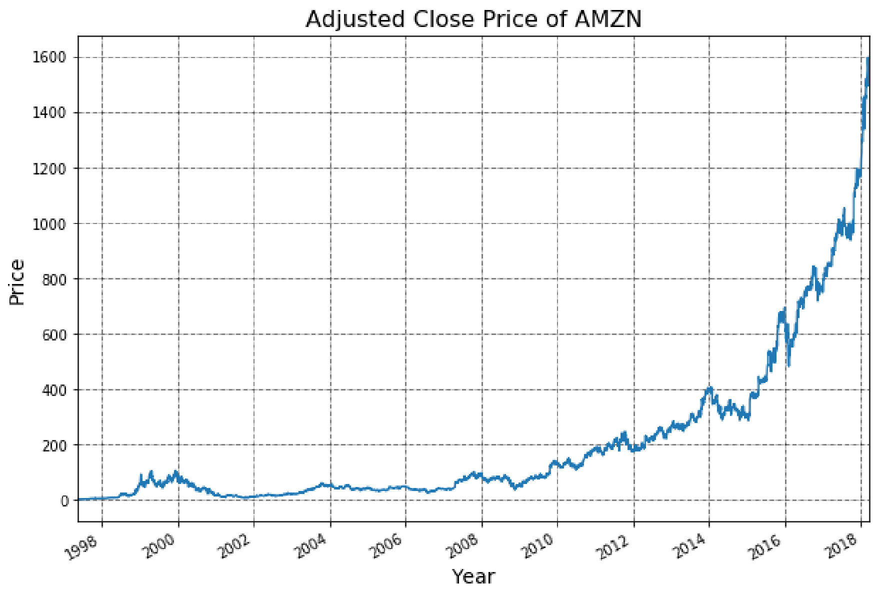
A financial dashboard offers all of the data, metrics, and insights needed to ensure the success of your financial performance, cash flow, cash management, and profit and loss analysis. The financial graph example above, associated with our business dashboard not only makes extracting key data swiftly but is developed in a way that makes communicating your findings to important stakeholders within the business far more simple. And in contrast to a traditional Excel chart, these financial graphs serve real-time data that will prove invaluable to the financial future of your business.

Downloading Datasets with Python

Downloads images from the Google Landmarks dataset using multiple threads. Images that already exist will not be downloaded again, so the script can resume a partially completed download. All images will be saved in the JPG format with 90% compression quality.

In [1]:

```
# Define the figure size for the plot
plt.figure(figsize=(10, 7))
# Plot the adjusted close price
data['Adj. Close'].plot()
# Define the label for the title of the figure
plt.title("Adjusted Close Price of %s" % ticker, fontsize=16)
# Define the labels for x-axis and y-axis
plt.ylabel('Price', fontsize=14)
plt.xlabel('Year', fontsize=14)
# Plot the grid lines
plt.grid(which="major", color='k', linestyle='-.', linewidth=0.5)
plt.show()
```



Get stock market data for multiple tickers

To get the stock market data of multiple stock tickers, you can create a list of tickers and call the quandl get method for each stock ticker.[1]

For simplicity, I have created a dataframe data to store the adjusted close price of the stocks.

In [4]:

```
# Define the ticker list
import pandas as pd
tickers_list = ['AAPL', 'IBM', 'MSFT', 'WMT']
# Import pandas
data = pd.DataFrame(columns=tickers_list)
# Feth the data
for ticker in tickers_list:
 data[ticker] = quandl.get('WIKI/' + ticker, start_date=start_date,
 end_date=end_date, api_key=QUANDL_API_KEY)['Adj. Close']
```

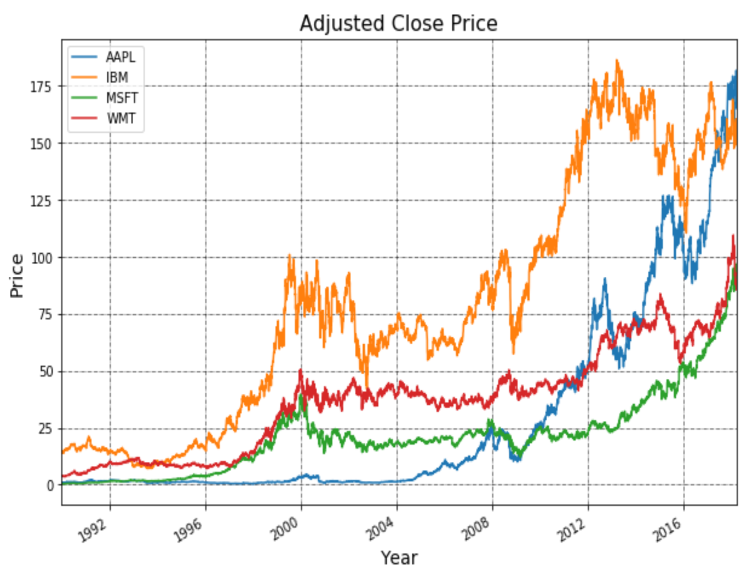
```
# Print first 5 rows of the data
data.head()
```

out[1]

Date	**AAPL**	**IBM**	**MSFT**	**WMT**
1990-01-02	1.118093	14.138144	0.410278	4.054211
1990-01-03	1.125597	14.263656	0.412590	4.054211
1990-01-04	1.129499	14.426678	0.424702	4.033561
1990-01-05	1.133101	14.390611	0.414300	3.990541
1990-01-08	1.140605	14.480057	0.420680	4.043886

In [2]:

```
# Plot all the close prices
data.plot(figsize=(10, 7))
# Show the legend
plt.legend()
# Define the label for the title of the figure
plt.title("Adjusted Close Price", fontsize=16)
# Define the labels for x-axis and y-axis
plt.ylabel('Price', fontsize=14)
plt.xlabel('Year', fontsize=14)
# Plot the grid lines
plt.grid(which="major", color='k', linestyle='-.', linewidth=0.5)
plt.show()
```



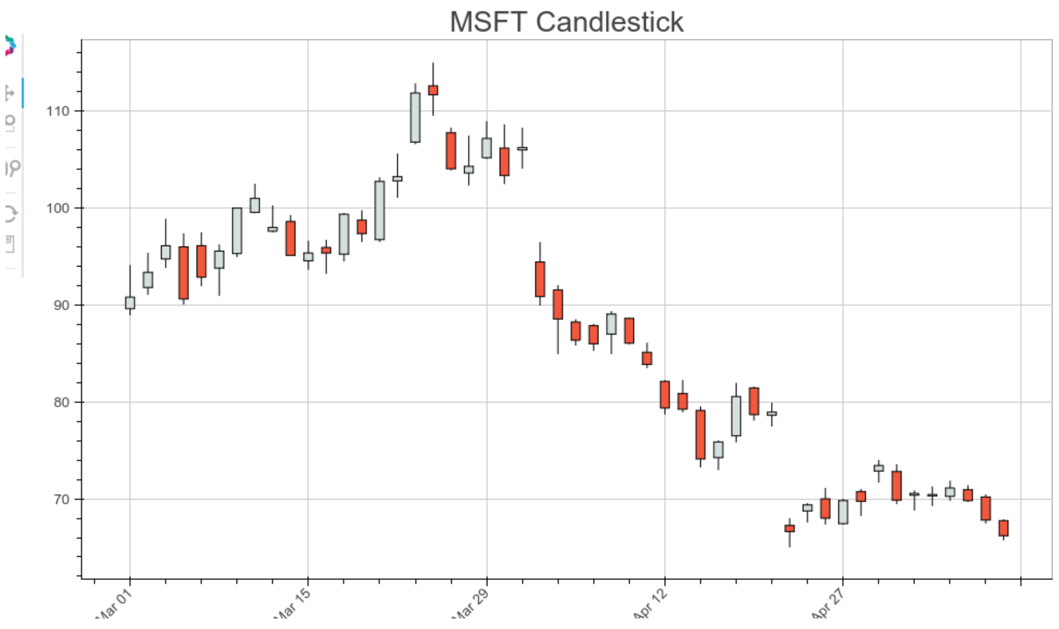
```
In [1]:
# Import the quandl
import quandl
# To get your API key, sign up for a free Quandl account.
# Then, you can find your API key on Quandl account settings page.
QUANDL_API_KEY = 'REPLACE-THIS-TEXT-WITH-A-REAL-API-KEY'
# This is to prompt you to change the Quandl Key
if QUANDL_API_KEY == 'REPLACE-THIS-TEXT-WITH-A-REAL-API-KEY':
 raise Exception("Please provide a valid Quandl API key!")
# Set the start and end date
start_date = '1990-01-01'
end_date = '2018-03-01'
# Set the ticker name
ticker = 'AMZN'
# Feth the data
data = quandl.get('WIKI/'+ticker, start_date=start_date,
 end_date=end_date, api_key=QUANDL_API_KEY)
```

```
# Print the first 5 rows of the dataframe
Data.
```

Candlestick Charts with Bokeh Quadrants

```
from math import pi

import pandas as pd

from bokeh.plotting import figure, show, output_file
from bokeh.sampledata.stocks import MSFT

df = pd.DataFrame(MSFT)[:50]
df["date"] = pd.to_datetime(df["date"])

mids = (df.open + df.close)/2
spans = abs(df.close-df.open)

inc = df.close > df.open
dec = df.open > df.close
w = 12*60*60*1000 # half day in ms

TOOLS = "pan,wheel_zoom,box_zoom,reset,save"

p=figure(x_axis_type="datetime",tools=TOOLS,plot_width=1000,toolbar_location="le
ft")

p.title = "MSFT Candlestick"
```

```
p.xaxis.major_label_orientation = pi/4
p.grid.grid_line_alpha=0.3

p.segment(df.date, df.high, df.date, df.low, color="black")
p.rect(df.date[inc],mids[inc],w, spans[inc], fill_color="#D5E1DD", line_color="black")
p.rect(df.date[dec],mids[dec], w, spans[dec], fill_color="#F2583E", line_color="black")

output_file("candlestick.html", title="candlestick.py example")
```

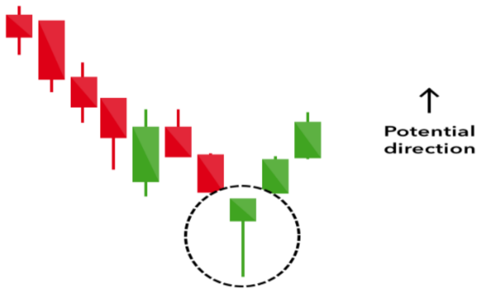


Candlestick Segments

When using any candlestick pattern, it is important to remember that although they are great for quickly predicting trends, they should be used alongside other forms of technical analysis to confirm the overall trend.

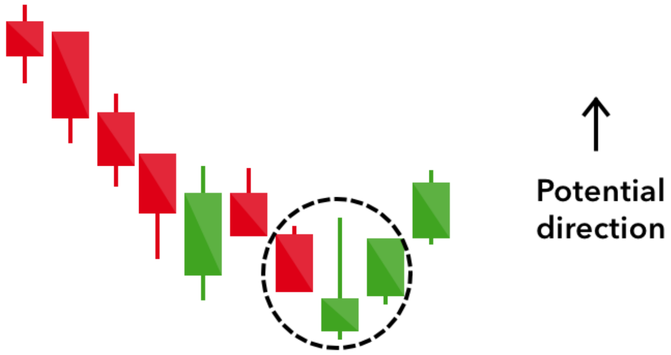
Hammer

The hammer candlestick pattern is formed of a short body with a long lower wick, and is found at the bottom of a downward trend.



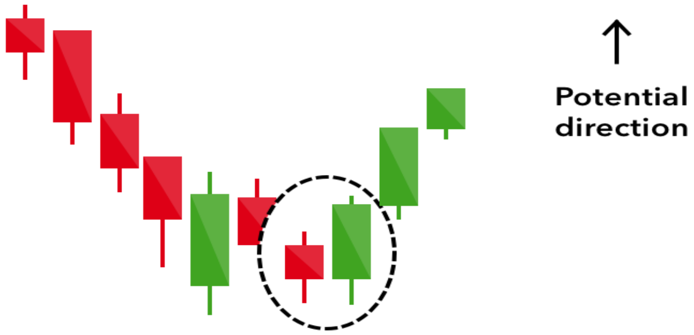
Inverse hammer

A similarly bullish pattern is the inverted hammer. The only difference being that the upper wick is long, while the lower wick is short.



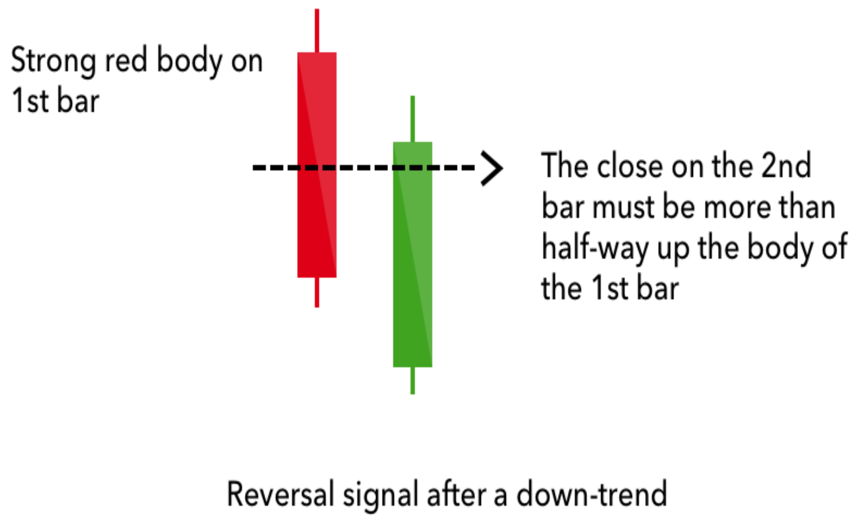
Bullish engulfing

The bullish engulfing pattern is formed of two candlesticks. The first candle is a short red body that is completely engulfed by a larger green candle.



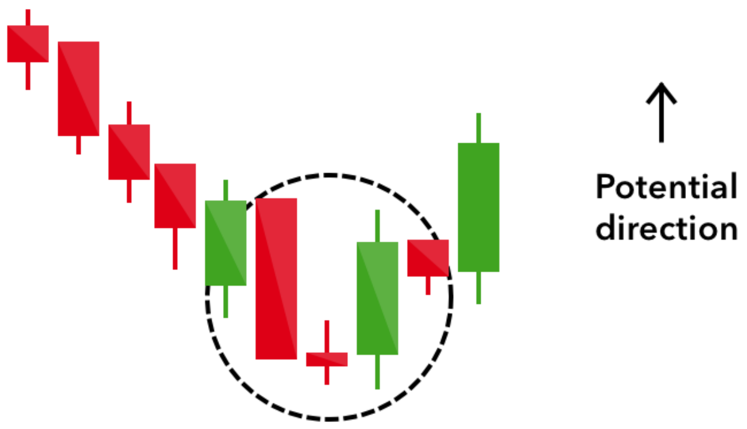
Piercing line

The piercing line is also a two-stick pattern, made up of a long red candle, followed by a long green candle.


Strong red body on 1st bar
The close on the 2nd bar must be more than half-way up the body of the 1st bar
Reversal signal after a down-trend

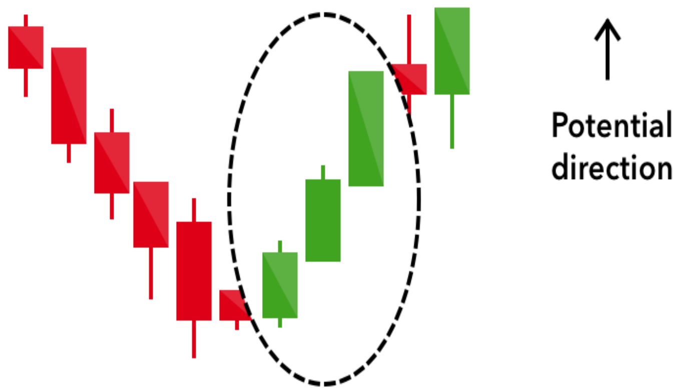
Morning star

The morning star candlestick pattern is considered a sign of hope in a bleak market downtrend. It is a three-stick pattern: one short-bodied candle between a long red and a long green. Traditionally, the 'star' will have no overlap with the longer bodies, as the market gaps both on open and close.



Three white soldiers

The three white soldiers pattern occurs over three days. It consists of consecutive long green (or white) candles with small wicks, which open and close progressively higher than the previous day.



The Concept Behind Embedding Bokeh Charts in a Flask Webpage

Installing Bokeh and Flask

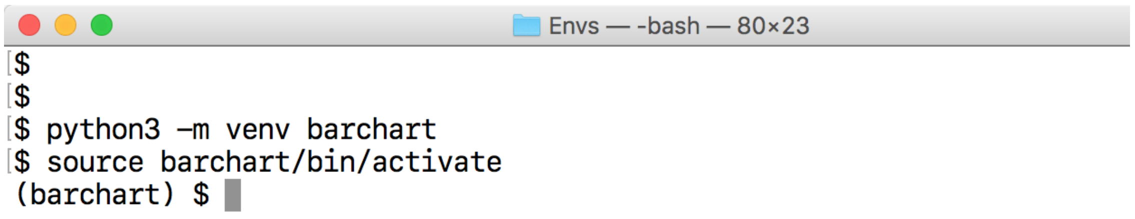
Create a fresh virtual environment for this project to isolate our dependencies using the following command in the terminal. I typically run this command within a separate venvs directory where all my virtualenvs are store.

python3 -m venv barchart

Activate the virtualenv.

source barchart/bin/activate

The command prompt will change after activating the virtualenv:



```
$
$
$ python3 -m venv barchart
$ source barchart/bin/activate
 (barchart) $
```

Activating our Python virtual environment on the command line.

Bokeh and Flask are installable into the now-activated virtualenv using pip. Run this command to get the appropriate Bokeh and Flask versions.

pip install bokeh==0.12.5 flask==0.12.2 pandas==0.20.1

After a brief download and installation period our required dependencies should be installed within our virtualenv. Look for output to confirm everything worked.

```
Installing collected packages: six, requests, PyYAML, python-dateutil, MarkupSafe, Jinja2, numpy, tornado, bokeh, Werkzeug, itsdangerous, click, flask, pytz, pandas
  Running setup.py install for PyYAML ... done
  Running setup.py install for MarkupSafe ... done
  Running setup.py install for tornado ... done
  Running setup.py install for bokeh ... done
  Running setup.py install for itsdangerous ... done
Successfully installed Jinja2-2.9.6 MarkupSafe-1.0 PyYAML-3.12 Werkzeug-0.12.2 bokeh-0.12.5 click-6.7 flask-0.12.2 itsdangerous-0.24 numpy-1.12.1 pandas-0.20.1 python-dateutil-2.6.0 pytz-2017.2 requests-2.14.2 six-1.10.0 tornado-4.5.1
```

Now we can start building our web application.

Starting Our Flask App

We are going to first code a basic Flask application then add our bar chart to the rendered page.

Create a folder for your project then within it create a file named app.py with these initial contents:

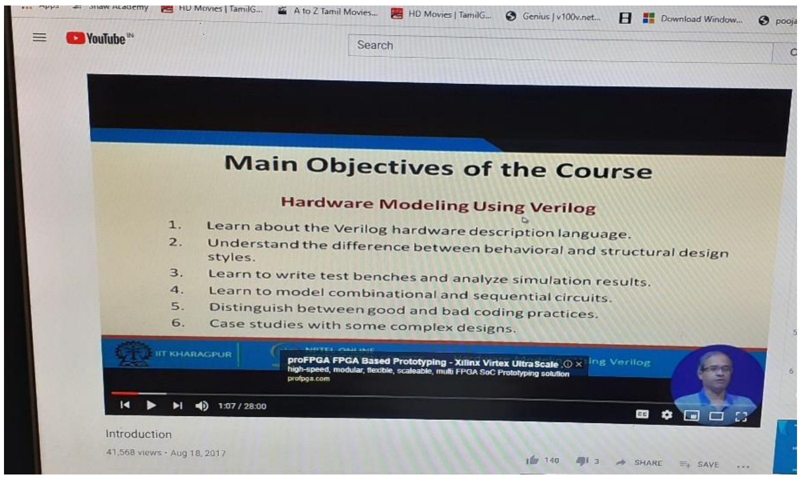
```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/<int:bars_count>/")
def chart(bars_count):
    if bars_count <= 0:
        bars_count = 1
    return render_template("chart.html", bars_count=bars_count)
if __name__ == "__main__":
    app.run(debug=True)
```

The above code is a short one-route Flask application that defines the chart function. chart takes in an arbitrary integer as input which will later be used to define how much data we want in our bar chart. The render_template function within chart will use a template from Flask's default template engine named Jinja2 to output HTML.

AFTERNOON SESSION DETAILS

Date:	4-6-2020	Name:	Kavyashree m
Course:	DIGITAL DESIGN USING HDL	USN:	4al15ec036
Topic:	Hardware modelling using Verilog , FPGA and ASII Interview questions	Semester & Section:	8th A
Github Repository:	Kavya		

Image of session



Main Objectives of the Course

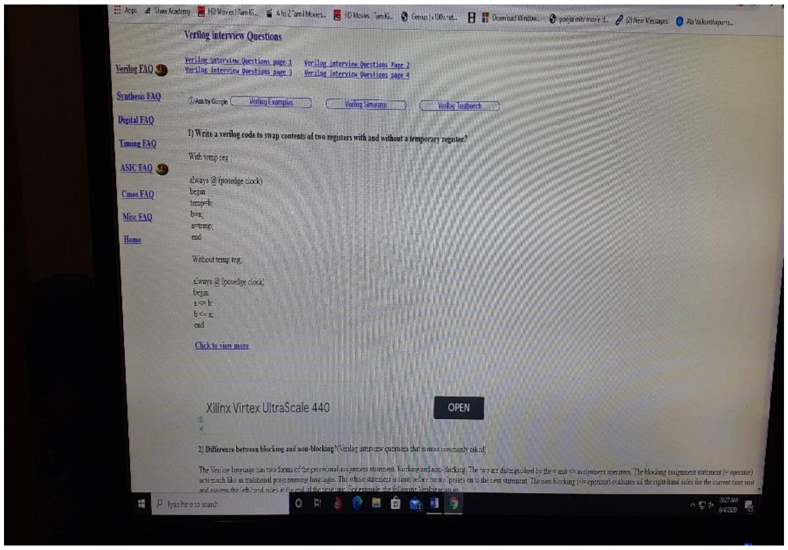
Hardware Modeling Using Verilog

1. Learn about the Verilog hardware description language.
2. Understand the difference between behavioral and structural design styles.
3. Learn to write test benches and analyze simulation results.
4. Learn to model combinational and sequential circuits.
5. Distinguish between good and bad coding practices.
6. Case studies with some complex designs.

Introduction

41,568 views • Aug 18, 2017

Fig 1: Hardware modelling using Verilog



Verilog interview Questions

Verilog interview Questions page 1 Verilog interview Questions Page 2
Verilog interview Questions page 3 Verilog interview Questions page 4

1) Write a verilog code to swap contents of two registers with and without a temporary register?

With temp reg :

```
always @ (posedge clock)
begin
temp=b;
b=a;
a=temp;
end
```

Without temp reg:

```
always @ (posedge clock)
begin
a <= b;
b <= a;
end
```

Click to view more

Xilinx Virtex UltraScale 440

2) Difference between blocking and non-blocking?(Verilog interview questions that is most commonly asked)

The Verilog language has two forms of the procedural assignment statement: blocking and non-blocking. The two are distinguished by the = and <= assignment operators. The blocking assignment statement (= operator) acts much like in traditional programming languages. The whole statement is done before control passes on to the next statement. The non-blocking (<= operator) evaluates all the right-hand sides for the current time unit and assigns the left-hand sides at the end of the time unit. For example, the following Verilog program

Fig 2 : FPGA and ASIC Interview questions

Hardware modelling using Verilog

Structure of HDL

HDLs are standard text-based expressions of the structure of electronic systems and their behaviour over time. Like concurrent programming languages, HDL syntax and semantics include explicit notations for expressing concurrency. However, in contrast to most software programming languages, HDLs also include an explicit notion of time, which is a primary attribute of hardware. Languages whose only characteristic is to express circuit connectivity between a hierarchy of blocks are properly classified as netlist languages used in electric computer-aided design. HDL can be used to express designs in structural, behavioral or register-transfer-level architectures for the same circuit functionality; in the latter two cases the synthesizer decides the architecture and logic gate layout.

HDL and programming languages

An HDL is grossly similar to a software programming language, but there are major differences. Most programming languages are inherently procedural (single-threaded), with limited syntactical and semantic support to handle concurrency. HDLs, on the other hand, resemble concurrent programming languages in their ability to model multiple parallel processes that automatically execute independently of one another. Any change to the process's input automatically triggers an update in the simulator's process stack.

Both programming languages and HDLs are processed by a compiler (often called a synthesizer in the HDL case), but with different goals. For HDLs, "compiling" refers to logic synthesis; the process of transforming the HDL code listing into a physically realizable gate netlist. The netlist output can take any of many forms: a "simulation" netlist with gate-delay information, a "handoff" netlist for post-synthesis placement and routing on a semiconductor die, or a generic industry-standard Electronic Design Interchange Format.

On the other hand, a software compiler converts the source-code listing into a microprocessor-specific object code for execution on the target microprocessor. As HDLs and programming languages borrow concepts and features from each other, the boundary between them is becoming less distinct. However, pure HDLs are unsuitable for general purpose application software development, just as general-purpose programming languages are undesirable for modeling hardware.

Yet as electronic systems grow increasingly complex, and reconfigurable systems become increasingly common, there is growing desire in the industry for a single language that can perform some tasks of both hardware design and software programming. SystemC is an example of such—embedded system hardware can be modeled as non-detailed architectural blocks.The target application is written in C or C++ and natively compiled for the host-development system; as opposed to targeting the embedded CPU, which requires host-simulation of the embedded CPU or an emulated CPU.

The high level of abstraction of SystemC models is well suited to early architecture exploration, as architectural modifications can be easily evaluated with little concern for signal-level implementation issues. However, the threading model used in SystemC relies on shared memory, causing the language not to handle parallel execution or low-level models well.

FPGA and ASIC Interview questions

What is FPGA ?

A field-programmable gate array is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnect.

What do conditional assignments get inferred into?

Conditionals in a continuous assignment are specified through the “?:” operator. Conditionals get inferred into a multiplexo.

What value is inferred when multiple procedural assignments made to the same reg variable in an always block?

When there are multiple nonblocking assignments made to the same reg variable in a sequential always block, then the last assignment is picked up for logic synthesis.

What is minimum and maximum frequency of dcm in spartan-3 series fpga?

Spartan series dcm’s have a minimum frequency of 24 MHZ and a maximum of 248

What are different types of FPGA programming modes?what are you currently using ?how to change from one to another?

Before powering on the FPGA, configuration data is stored externally in a PROM or some other nonvolatile medium either on or off the board. After applying power, the configuration data is written to the FPGA using any of five different modes: Master Parallel, Slave Parallel, Master Serial, Slave Serial, and Boundary Scan (JTAG).The Master and Slave Parallel modes Mode selecting pins can be set to select the mode, refer data sheet for further details.

Can you explain what struck at zero means?

These stuck-at problems will appear in ASIC. Some times, the nodes will permanently tie to 1 or 0 because of some fault. To avoid that, we need to provide testability in RTL.If it is permanently 1 it is called stuck-at-1.If it is permanently0itiscalledstuck-at-0.

Task-4

Implement a simple T Flipflop and test the module using a compiler

Design

```
module tff (   input clk,
              input rstn,
              input t,
            output reg q);

  always @ (posedge clk) begin
    if (!rstn)
      q <= 0;
    else
      if (t)
          q <= ~q;
      else
          q <= q;
  end
endmodule
```

Testbench

```
module tb;
  reg clk;
  reg rstn;
  reg t;

  tff u0 (  .clk(clk),
            .rstn(rstn),
            .t(t),
          .q(q));

  always #5 clk = ~clk;

  initial begin
    {rstn, clk, t} <= 0;

    $monitor ("T=%0t rstn=%0b t=%0d q=%0d", $time, rstn, t, q);
    repeat(2) @(posedge clk);
    rstn <= 1;

    for (integer i = 0; i < 20; i = i+1) begin
      reg [4:0] dly = $random;
      #(dly) t <= $random;
    end
  #20 $finish;
  end
endmodule
```

Simulation Log

```
ncsim> run
T=0 rstn=0 t=0 q=x
T=5 rstn=0 t=0 q=0
T=15 rstn=1 t=0 q=0
T=19 rstn=1 t=1 q=0
```

```
T=25 rstn=1 t=1 q=1

T=35 rstn=1 t=1 q=0

T=43 rstn=1 t=0 q=0

T=47 rstn=1 t=1 q=0

T=55 rstn=1 t=0 q=1

T=59 rstn=1 t=1 q=1

T=65 rstn=1 t=1 q=0

T=67 rstn=1 t=0 q=0

T=71 rstn=1 t=1 q=0

T=75 rstn=1 t=0 q=1

T=79 rstn=1 t=1 q=1

T=83 rstn=1 t=0 q=1

T=87 rstn=1 t=1 q=1

T=95 rstn=1 t=0 q=0

Simulation complete via $finish(1) at time 115 NS + 0
```