

DAILY ASSESSMENT FORMAT

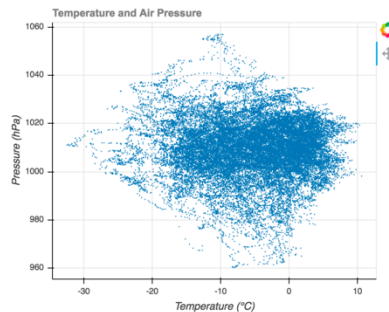
Date:	2/06/2020	Name:	K B KUSHI
Course:	Python	USN:	4AL17EC107
Topic:	<ul style="list-style-type: none"> Interactive Data Visualization with Bokeh Webscraping with Python Beautiful Soup 	Semester & Section:	6 B
Github Repository:	https://github.com/alvas-education-foundation/KUSHI-COURSES.git		

FORENOON SESSION DETAILS

Image of session

Plotting Weather Data (Practice)

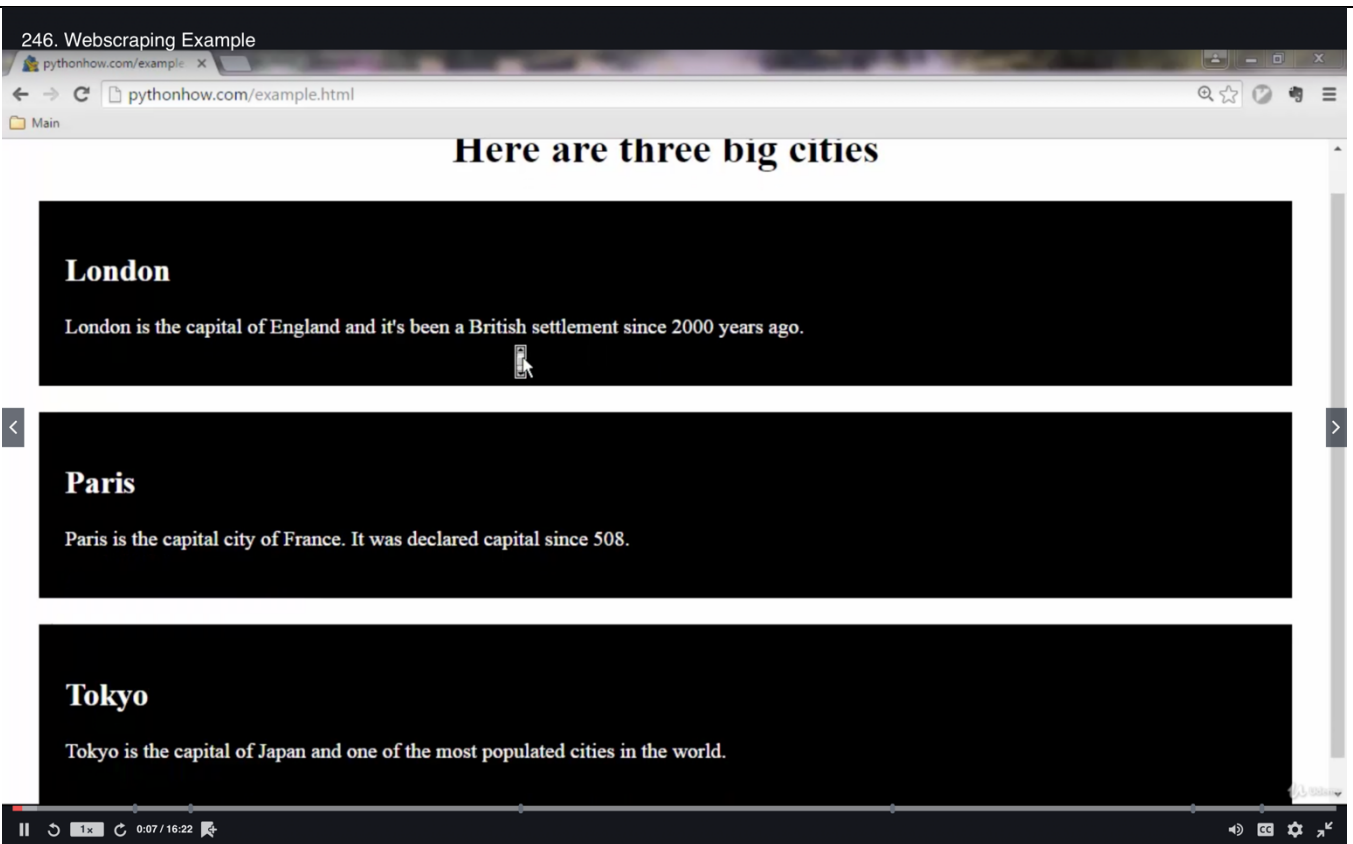
Produce the following graph using the data from this Excel file: <http://pythonhow.com/data/verlegenuken.xlsx>



Some notes:

Temperature and pressure values in the Excel file have a scale factor of 10; you'll have to divide those values by 10 to get the actual observations.

And, yes, you can set your own fonts and colors, but be accurate with the rest of the plot elements.



Report:

1. Interactive Data Visualization with Bokeh:

- Bokeh is a data visualization library in Python that provides high-performance interactive charts and plots.
- It is possible to embed bokeh plots in Django and flask apps.
- Bokeh provides two visualization interfaces to users: bokeh models :
- A low level interface that provides high flexibility to application developers.

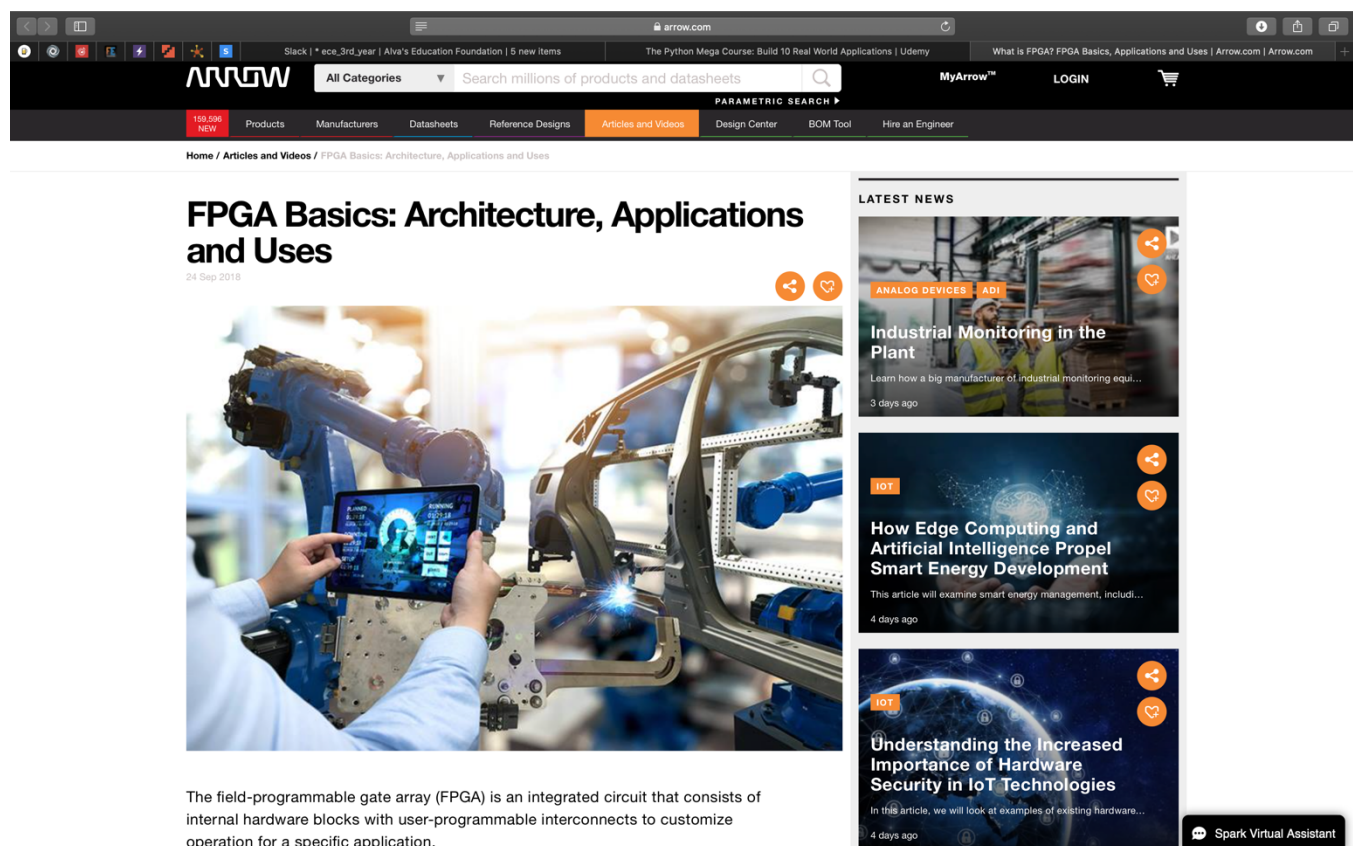
2. Webscraping with Python Beautiful Soup

- Web scraping is a term used to describe the use of a program or algorithm to extract and process large amounts of data from the web.
- Whether you are a data scientist, engineer, or anybody who analyzes large amounts of datasets, the ability to scrape data from the web is a useful skill to have.
- Let's say you find data from the web, and there is no direct way to download it, web scraping using.
- Python is a skill you can use to extract the data into a useful form that can be imported.

Date:	2/06/2020	Name:	K B KUSHI
Course:	DIGITAL DESIGN USING HDL	USN:	4AL17EC107
Topic:	<ol style="list-style-type: none"> FPGA Basics: Architecture, Applications and Uses Verilog HDL Basics by Intel Verilog Testbench code to verify the design under test (DUT) 	Semester & Section:	6 B
Github Repository:	https://github.com/alvas-education-foundation/KUSHI-COURSES.git		

AFTERNOON SESSION DETAILS

Image of session



The screenshot shows the Arrow.com website. The main article is titled "FPGA Basics: Architecture, Applications and Uses" and is dated 24 Sep 2018. The article features a large image of a person in a lab coat using a tablet to interact with a robotic arm in a factory setting. Below the image, the text reads: "The field-programmable gate array (FPGA) is an integrated circuit that consists of internal hardware blocks with user-programmable interconnects to customize operation for a specific application."

On the right side of the page, there is a "LATEST NEWS" section with three articles:

- Industrial Monitoring in the Plant** (Analog Devices, ADI) - 3 days ago
- How Edge Computing and Artificial Intelligence Propel Smart Energy Development** (IoT) - 4 days ago
- Understanding the Increased Importance of Hardware Security in IoT Technologies** (IoT) - 4 days ago

At the bottom right, there is a "Spark Virtual Assistant" button.

Report – Report can be typed or hand written for up to two pages.

FPGA Design

How do we transform this collection of thousands of hardware blocks into the correct configuration to execute the application? An FPGA-based design begins by defining the required computing tasks in the development tool, then compiling them into a configuration file that contains information on how to hook up the CLBs and other modules. The process is similar to a software development cycle except that the goal is to architect the hardware itself rather than a set of instructions to run on a predefined hardware platform. Designers have traditionally used a hardware description language (HDL) such as VHDL or Verilog to design the FPGA configuration.

CPLD vs FPGA

Originally, FPGAs included the blocks in Figure 1 and little else, but now designers can choose from products with a large range of features. Less complex devices such as simple programmable logic devices (SPLDs) and complex programmable logic devices (CPLDs) bridge the gap between discrete logic devices and entry-level FPGAs.

Entry-level FPGAs emphasize low power consumption, low logic density and low complexity per chip. Higher-function devices add functional blocks dedicated to specific functions: Examples include clock management components, phase-locked loops (PLLs), high-speed serializers and deserializers, Ethernet MACs, PCI express controllers and high-speed transceivers. These blocks can either be implemented with CLBs—termed soft IP—or designed as separate circuits; i.e., hard IP. Hard IP blocks gain performance at the expense of reconfigurability.

At the high end, the FPGA product family includes complex system-on-chip (SoC) parts that integrate the FPGA architecture, hard IP and a microprocessor CPU core into a single component. Compared to separate devices, a SoC FPGA provides higher integration, lower power, smaller board size and higher-bandwidth communication between the core and other blocks.

Implement a 4:1 MUX and write the test bench code to verify the module

Verilog design

```
module mux41(  
    input i0,i1,i2,i3,sel0,sel1,  
    output reg y);  
  
    always @(*)  
    begin  
        case ({sel0,sel1})  
            2'b00 : y = i0;  
            2'b01 : y = i1;  
            2'b10 : y = i2;  
            2'b11 : y = i3;  
        endcase  
    end  
  
endmodule
```

TestBench

```
module tb_mux41;  
  
    reg i0,i1,i2,i3,SEL0,SEL1;  
    wire Y;
```

```
mux41 MUX (.i0(I0),.i1(I1),.i2(I2),.i3(I3),.sel0(SEL0),.sel1(SEL1),.y(Y));
```

```
initial begin
```

```
    I0 =1'b0;
```

```
    I1= 1'b0;
```

```
    I2 =1'b0;
```

```
    I3 =1'b0;
```

```
    SEL0 =1'b0;
```

```
    SEL1 =1'b0;
```

```
    #45 $finish;
```

```
end
```

```
always #2 I0 = ~I0;
```

```
always #4 I1 =~I1;
```

```
always #6 I2 =~I1;
```

```
always #8 I3 =~I1;
```

```
always #3 SEL0 = ~SEL0;
```

```
always #3 SEL1 = ~SEL1;
```

```
always @(Y)
```

```
    $display( "time =%0t INPUT VALUES: \t I0=%b I1 =%b I2 =%b I3 =%b SEL0 =%b SEL1 =%b \t output value Y  
    =%b ",$time,I0,I1,I2,I3,SEL0,SEL1,Y);
```

```
endmodule
```

OUTPUT

```
time =0 INPUT VALUES:  I0=0 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0      output value Y =0
```

```
time =2 INPUT VALUES:  I0=1 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0      output value Y =1
```

```
time =3 INPUT VALUES:  I0=1 I1 =0 I2 =0 I3 =0 SEL0 =1 SEL1 =1      output value Y =0
```

time =6 INPUT VALUES: I0=1 I1 =1 I2 =0 I3 =0 SEL0 =0 SEL1 =0 output value Y =1

time =8 INPUT VALUES: I0=0 I1 =0 I2 =0 I3 =0 SEL0 =0 SEL1 =0 output value Y =0

time =14 INPUT VALUES: I0=1 I1 =1 I2 =1 I3 =0 SEL0 =0 SEL1 =0 output value Y =1

time =15 INPUT VALUES: I0=1 I1 =1 I2 =1 I3 =0 SEL0 =1 SEL1 =1 output value Y =0