

uvm_tb_arch_doc_py

Table of contents:

1. Introduction

2. Overview of the Modules

3. Description of each Module

3.1 TB_arch_using_image_draw.py

3.2 component_name_finder.py

3.3 draw_rect_using_turtle.py

3.4 draw_rect_with_text_inside.py

3.5 pattern_finder_indir.py

3.6 tb_arch_img_draw.py

3.7 uvm_tb_arch_agent.py

3.8 uvm_tb_name_image_file_generator.py

1. Introduction

uvm_tb_arch_doc_py is a python project to automatically generates the UVM testbench Architecture

The main aim of this project is, by using the Python Programming language we have to generate the UVM testbench Architecture template. For generating the UVM TB Architecture we have to write an example testbench code (top, test, env, agent etc) in UVM Methodology.

For generating the UVM Testbench template, by using Python Programming we can use either Python Turtle graphics or Python Image draw graphics

- Python turtle Graphics is used to create shapes and patterns needs to import python turtle from python library, by using Python turtle can generate different shapes but before going to use this Turtle library methods initially we have to define these methods it makes the Turtle graphics as inefficient
- The Image Draw module provides simple 2D graphics support for Image Object. These graphics interface uses the same coordinate system as PIL. PIL is the Python Imaging Library which provides the python interpreter with Image Editing Capabilities. It is simpler and easier to understand it can be install by pip(pip install Pillow) . The pillow color schemes we use is RGB. The color RGB representation and support is provided by the module **ImageColor**.

How to Run python files

Step1: Download the latest python version in Desktop

Step2: Get Pycharm tool or can use command prompt window

Step3: Once the Pycharm tool installed in Desktop, go to file create a project

Step4: PyCharm enables programmers to write high-quality Python code in code editor. The editor enables programmers to read code easily through color schemes. Write an example code in the code editor and then save the file in the project

Step5: Right click on the file then select run option and execute the code

2. Overview of the Modules

Module	Description
dummy_tb	<p>In this folder, simple Memory testbench module code is written in UVM Methodology.</p> <p>Components that are used for building the Memory testbench module are mem_seq_item, mem_wr_seq, mem_rd_seq, mem_seqr, mem_drv, mem_mon, mem_agent, mem_env, mem_test, mem_top.</p>
TB_arch_using_image_draw.py	<p>This TB_arch_using_image_draw script module defines ,</p> <ul style="list-style-type: none">• How the TB Architecture Blocks (like top, test, env, dut, scoreboard, interface, virtual interface, agent, monitor, driver, seqr) should be generate in the image format with respect to given input co-ordinates && colours• It also defines, how the connections are happened between the Driver to Virtual Interface, Virtual interface to Interface, Interface to DUT and Virtual interface to Driver
component_name_finder.py	<p>This component_name_finder script modules defines,</p> <ul style="list-style-type: none">• When the user wants to find particular component in the TB Architecture, first user have to enter the input as the TB Directory path and then enter the Keyword of the particular component name which want to search• Then the Output of this script generates the user selected TB component class name and prints the output in Normal && Table format
draw_rect_using_turtle.py	<p>This draw_rect_using_turtle script module defines</p> <ul style="list-style-type: none">• how to draw the rectangle shape by using Turtle library

Module	Description
draw_rect_with_text_inside.py	<p>This draw_rect_with_text_inside scripting module defines</p> <ul style="list-style-type: none"> • How to write the text inside the rectangle shape by using Image Draw module
pattern_finder_indir.py	<p>This pattern_finder_indir Script defines,</p> <ul style="list-style-type: none"> • How to find a pattern in all files of given directory i.e., how to find the UVM components from the given testbench directory • Then it prints the filename and the path of the file. This can be used to find all the different components of a testbench.
skeleton of component search python	<p>This skeleton_component_search script defines</p> <ul style="list-style-type: none"> • How to find the generic word in TB components that can be searched to determine the presence of component in TB
tb_arch_img_draw.py	<p>This tb_arch_img_draw script defines by using Image, Image draw Module</p> <ul style="list-style-type: none"> • How to draw the agent components by reading the tb_info.txt file, these file which contains the information of agent i.e., no of agents
uvm_tb_arch_agent.py	<p>This uvm_tb_arch_agent script defines</p> <ul style="list-style-type: none"> • How to support number of agents upto 7 in the TB Architecture

Module	Description
uvm_tb_name_image_file_generator.py	<p>This uvm_tb_name_image_file_generator script defines</p> <ul style="list-style-type: none"> • Here user enters the path of TB directory. Setting the width and height according to the screen size and Setting the canvas size, create lookup table for uvm component and name. • Create file for writing component and component name, also clean file before writing the data. <ul style="list-style-type: none"> ➤ Drawing top level structure, set the co-ordinates and set color for top as yellow color. ➤ Drawing test level structure, set the co-ordinates and set color for top as orange color. ➤ Drawing env level structure, set the co-ordinates and set color for top as yellow color. ➤ Drawing scoreboard level structure, set the co-ordinates and set color for top as red. ➤ Drawing agent level structure, set the co-ordinates and set color for top as blue color also set the image font. ➤ Calling the rectangular creation function. Here, user have to enter the input TB directory path then it will create the block diagram of component TB Architecture like-top, test, env, agent, scoreboard

3. Description of each Module

3.1 TB_arch_using_image_draw.py

- To draw TB Diagram aimed at 2 agentsTo draw Rectangle with given co-ordinate & fill with given colour and write the text inside rectangle
- To draw top, test, env blocks write the value of n choosen height of env block should be less , so we have to give proper dimensions
- This “docx” module is to manipulate with docs like MS Word. Used it to add TB diagram to the document, we have take handle doc for docx
- After we have to import tkinter & setting height & width to measure whole screen size and then
 - Create 1st outer rectangle for top
 - Create 2nd inner rectangle for test
 - Create 3rd inner rectangle for env
 - Create 5th inner rectangle for scoreboard
 - Create 4th inner rectangle for sequences, DUT, Interface, Virtual Interface
 - Check for No of agents user have to give
 - Start another rectangle MON inside agent
 - Start another rectangle DRV inside agent
 - Start another rectangle SEQR inside agent
- Then draw the arrows between Driver to Virtual Interface, Virtual Interface to Interface and Driver1 to Virtual Interface and then set the start & end co-ordinates
- By using arrowed line() method set the colour & thickness and then add the picture & save the picture in docx


```

1 #####METHOD TO DRAW RECTANGLE WITH THE GIVEN CO-ORDINATES AND FILL WITH THE GIVEN COLOR#####
2 def draw_rect(image,coordinates,fill,color,width=1):
3     rect_start = (coordinates[0][0],coordinates[0][1]);
4     rect_end = (coordinates[1][0], coordinates [1][1])
5     image.rectangle((rect_start,rect_end),fill=fill,outline = color)
6 #Method to write the text inside the rectangle
7 def wr_text_in_rect(image,start_wr_w,start_wr_h,str,tfill):
8     font = ImageFont.truetype("arial.ttf", 16)
9     image.text((start_wr_w,start_wr_h),str, fill = tfill,font = font)
10 #Method to draw the top,test,env blocks (w.r.t. the value of n chosen)
11 def call_simple_rect(w,h,n,text,bfill,tfill,img1):
12     w1 = w - (n*10); #end of x should be max
13     if n != 5:
14         h2 = h - (n*10); #end of 'y' should be max
15         h1 = n*15 + 10;
16         w2 = h1;
17     elif n == 5: #The height of the env block should be less; So used like below dimensions
18         h2 = h - (n*10*5)
19         h1 = n*15 + 10 + 35;
20         w2 = n*15 + 10;
21     top_right = (w1,h1)
22     bottom_left = (w2,h2)
23     start_x = w1 - (50);
24     start_y = h1 + (n*2);
25     outline_width = 10
26     outline_color = "black"
27     draw_rect(img1,(top_right, bottom_left), fill=bfill ,color=outline_color, width=outline_width)
28     wr_text_in_rect(img1,start_x,start_y,text,tfill)
29     print ("Dimensions are %0d %0d %0d %0d",top_right, bottom_left)
30     return w1;
31 #This docx module is to manipulate with docs like MS Word. Used it to add TB diagram to the document
32 import docx
33 #This opencv module in python ease us to draw arrowed line in the image
34 import cv2
35 # This pillow module to import Image draw module
36 from PIL import Image,ImageDraw,ImageFont
37 #Taking the handle doc for docx
38 doc = docx.Document()
39 #This Module is used to measure the whole screen size

40 import tkinter
41 root = tkinter.Tk()
42 width = root.winfo_screenwidth()
43 height = root.winfo_screenheight()
44 print ("Width & HEIGHT",width,height)
45 # create line image of width and height
46 w = width
47 h = height
48 img = Image.new("RGB", (w, h),"white")
49 img1 = ImageDraw.Draw(img)
50 #Create first outer rectangle top n=1
51 n = 1;
52 top_dim = call_simple_rect(w,h,n,"TOP","orange","black",img1);
53 #Create second inner rectangle test n=3
54 n = 3;
55 test_dim = call_simple_rect(w,h,n,"TEST","pink","black",img1);
56 #Create third inner rectangle env n=5
57 n = 5;
58 env_dim = call_simple_rect(w,h,n,"ENV","yellow","black",img1);
59 #Create fifth inner rectangle SCOREBOARD
60 top_right = (w-140,150)
61 bottom_left = (400,230)
62 start_x = ((w-140)+400)/2 + 12;
63 start_y = 180;
64 draw_rect(img1,(top_right, bottom_left), fill="gray" ,color="black", width=10)
65 wr_text_in_rect(img1,start_x,start_y,"SCOREBOARD","BLACK")
66 #Create fourth inner rectangle sequences DUT
67 top_right = (90,h-80)
68 bottom_left = (w-40,h-50)
69 start_x_dut = (90 + (w-40))/2;
70 start_y_dut = (((h - 80) + (h - 50))/2) - 12;
71 draw_rect(img1,(top_right, bottom_left), fill="grey" ,color="black", width=10)
72 wr_text_in_rect(img1,start_x_dut,start_y_dut,"DUT","BLACK")
73 #Create fourth inner rectangle Interface
74 top_right = (90,h-120)
75 bottom_left = (w-40,h-150)
76 start_x_if = (90 + (w-40))/2;
77 start_y_if = ((h-120)+(h-150))/2 - 12;
78 draw_rect(img1,(top_right, bottom_left), fill="green" ,color="black", width=10)

```

```

79 wr_text_in_rect(img1,start_x_if,start_y_if,"INTERFACE","BLACK")
80 #Create fourth inner rectangle UIF
81 top_right = (90,h-190)
82 bottom_left = (w-40,h-220)
83 start_x_vif = (90 + (w-40))/2;
84 start_y_vif = ((h-190)+(h-220))/2 - 12;
85 draw_rect(img1,(top_right, bottom_left), fill="gray",color="black", width=10)
86 wr_text_in_rect(img1,start_x_vif,start_y_vif,"UIF","BLACK")
87 print(env_dim);
88 #Check for number of agents
89 n = 7;
90 agnt_cnt = int(input("Enter no. of agents"))
91 w1 = (env_dim/agnt_cnt)
92 h1 = (env_dim/agnt_cnt)
93 tx = 40;
94 x0 = 0;
95 diff = 0;
96 m1 = 1;
97 m2 = 0;
98 #To draw the number of agents w.r.t. agent count
99 for val in range(agnt_cnt):
100     print("VALUE OF X0 IS",x0)
101     x1 = tx + 55;
102     print("VALUE OF X1 IS",x1)
103     y0 = (n*4*10)
104     if agnt_cnt == 1:
105         x0 = (env_dim/agnt_cnt) - 20;
106     elif agnt_cnt != 1:
107         x0 = (m1*(env_dim/agnt_cnt))+(m2*(x1 + diff));
108     print("VALUE OF X0 LATER IS",x0)
109     y1 = h - (4*n*10)
110     tx = x0;
111     diff = x0 - x1;
112     top_right = (x0,y0);
113     bottom_left = (x1,y1);
114     start_x = x0 - 50;
115     start_y = y0 + 5;
116     outline_width = 10
117     print(top_right);
118     print(bottom_left);
119     draw_rect(img1,(top_right, bottom_left), fill="cyan",color="black", width=outline_width)
120     wr_text_in_rect(img1,start_x,start_y,"AGENT","BLACK")
121     #Start another rectangle MONITOR inside the agent
122     x3 = x1 + 10
123     y3 = y0 + 20
124     x2 = x0 - 350
125     y2 = y0 + 100
126     top_right = (x2,y2);
127     bottom_left = (x3,y3);
128     start_x_mon = (x2 + x3)/2;
129     start_y_mon = y3 + 15;
130     outline_width = 10
131     draw_rect(img1,(top_right, bottom_left), fill="orange",color="black", width=outline_width)
132     wr_text_in_rect(img1,start_x_mon,start_y_mon,"MONITOR","BLACK")
133     #Start another rectangle DRIVER inside the agent
134     x5 = x3 + 300
135     y5 = y3 + 100
136     x4 = x2 + 300
137     y4 = y2 + 100
138     top_right = (x4,y4);
139     bottom_left = (x5,y5);
140     start_x_drv = (x4 + x5)/2;
141     start_y_drv = y5 + 5;
142     outline_width = 10
143     draw_rect(img1,(top_right, bottom_left), fill="green",color="black", width=outline_width)
144     wr_text_in_rect(img1,start_x_drv,start_y_drv,"DRIVER","BLACK")
145     m2 = 1;
146     m1 = 0;
147     print("DRIVER",start_x_drv,start_y_drv)
148     #Start another rectangle SEQUENCER inside the agent
149     x5 = x3 + 300
150     y5 = y3
151     x4 = x2 + 300
152     y4 = y2
153     top_right = (x4,y4);
154     bottom_left = (x5,y5);
155     start_x_sqr = (x4 + x5)/2;
156     start_y_sqr = y5 + 5;

```

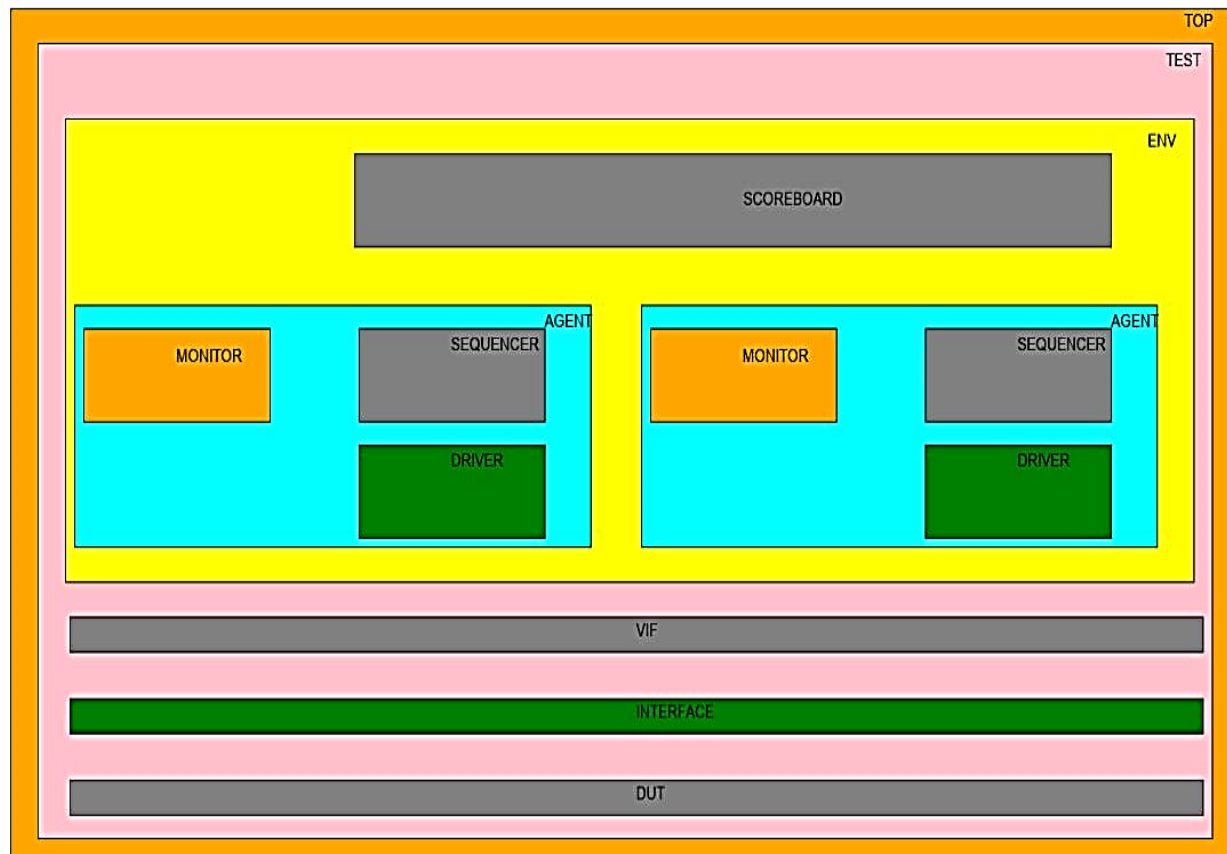
```

157     outline_width = 10
158     draw_rect(img1,(top_right, bottom_left), fill="grey" ,color="black", width=outline_width)
159     wr_text_in_rect(img1,start_x_sqr,start_y_sqr,"SEQUENCER","BLACK")
160     m2 = 1;
161     m1 = 0;
162 |   img.show()
163     img.save('E:\Python_task\tb_arch.jpg');
164     # Arrow Drawing
165     path = 'E:\Python_task\tb_arch.jpg'
166     # Reading an image in default mode
167     image = cv2.imread(path)
168     # Window name in which image is displayed
169     window_name = 'Image'
170     #####DRAW ARROW BETWEEN DRIVER AND VIF#####
171     start_point = (int(start_x_drv - 15),int(start_y_drv) + 75)
172     # End coordinate
173     end_point = (int(start_x_drv - 15),int(start_y_drv + 25) + 115)
174     color = (0, 0, 0)
175     thickness = 3
176     # Using cv2.arrowedLine() method
177     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
178     cv2.imshow(window_name, image)
179     cv2.imwrite(E:\Python_task\tb_arch.jpg",image)
180     #####DRAW ARROW BETWEEN VIF AND INTERFACE#####
181     # Start coordinate
182     start_point = (int(start_x_vif - 15),int(start_y_vif + 25))
183     # End coordinate
184     end_point = (int(start_x_if - 15),int(start_y_if - 5))
185     color = (0, 0, 0)
186     thickness = 3
187     # Using cv2.arrowedLine() method
188     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
189     cv2.imshow(window_name, image)
190     #####DRAW ARROW BETWEEN INTERFACE AND DUT#####
191     start_point = (int(start_x_if - 15),int(start_y_if + 25))
192     # End coordinate
193     end_point = (int(start_x_dut - 15),int(start_y_dut - 5))
194     color = (0, 0, 0)
195     thickness = 3
196     # Using cv2.arrowedLine() method
197     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
198     cv2.imshow(window_name, image)
199     #####DRAW ARROW BETWEEN DRIVER1 AND VIF#####
200     start_point = (506,482)
201 |   # End coordinate
202     end_point = (506,547)
203     color = (0, 0, 0)
204     thickness = 3
205     # Using cv2.arrowedLine() method
206     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
207     cv2.imshow(window_name, image)
208     cv2.imwrite("E:\Python_task\tb_arch.jpg",image)
209     doc.add_picture('E:\Python_task\tb_arch.jpg')
210     doc.save('E:\Python_task\pattern_printing_ex.docx')
...

```

Results of TB_arch_using_image_draw script

```
TB_arch_using_image_draw ×
D:\Python\Python392\python.exe E:/uvm_tb_arch_doc_py-main/TB_arch_using_image_draw.py
Width & HEIGHT 1366 768
Dimensions are %0d %0d %0d %0d (1356, 25) (25, 758)
Dimensions are %0d %0d %0d %0d (1336, 55) (55, 738)
Dimensions are %0d %0d %0d %0d (1316, 120) (85, 518)
1316
Enter no. of agents 2
VALUE OF X0 IS 0
VALUE OF X1 IS 95
VALUE OF X0 LATER IS 658.0
(658.0, 280)
(95, 488)
DRIVER 506.5 405
VALUE OF X0 IS 658.0
VALUE OF X1 IS 713.0
VALUE OF X0 LATER IS 1276.0
(1276.0, 280)
(713.0, 488)
DRIVER 1124.5 405
Process finished with exit code 0
```



3.2 component_name_finder.py

The component name finder script checks the particular keyword in all files all lines and collects the class name of components here user have to give the input directory path then it prints the output in normal and table format for further processing

Import the os, re, prettytable modules enter the testbench directory path, Open the doc file and then import the pretty table and assign the Class names of the Testbench code as a Keyword

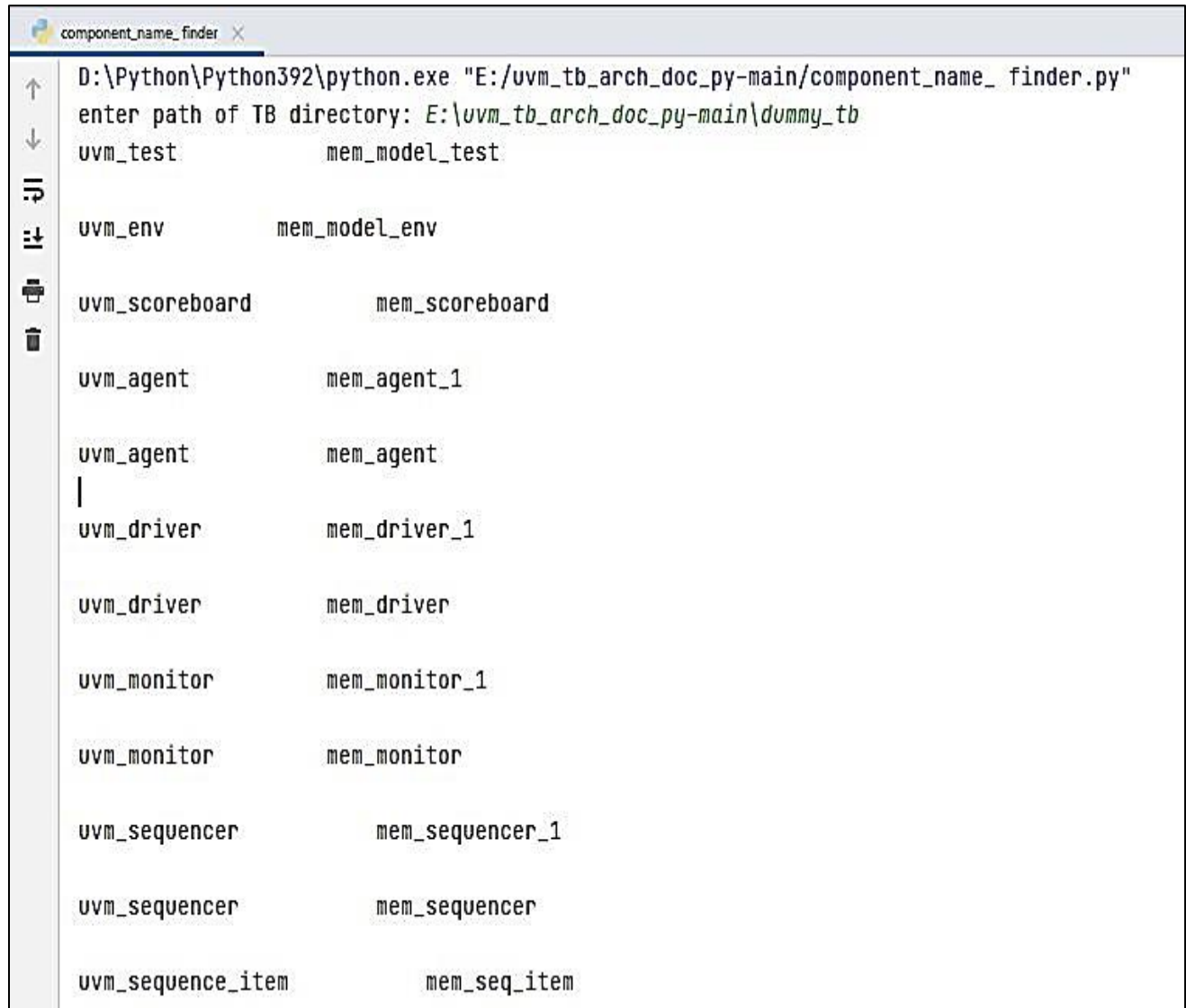
Then write the function for component search and then assign the Input directory path to root directory path. Here we have to call the component search function and checking the keyword of TB class names and opening the docx file write the information in the file and closing the file

```

1 import os, re, prettytable
2 path = input("enter path of TB directory: ") #user input for path of TB directory
3 open("tree_data.doc", "w+")
4 from prettytable import PrettyTable #to draw table of list
5 x = PrettyTable()
6 x.field_names = ["keyword", "class name"]
7 def component_search(keyword): # to search for keyword in all files of directory
8     root_dir = path
9     for root, dirs, files in os.walk(root_dir, onerror=None, topdown=True): # to loop inside all files of directory
10         for filename in files:
11             file_path = os.path.join(root, filename)
12             with open(file_path, "rb") as f: # read file as binary
13                 for line in f:
14                     line = line.decode("utf-8") #decode to string for read
15                     if keyword in line: # keyword determines the word to be looked into in each of the file
16                         #print(file_path,filename)
17                         #print(root)
18                         #print(root_dir)
19                         #print(files)
20                         component_name_finder(keyword, line) # call function to find class name
21                     break
22 def tb_comps(): #generic word in tb components that can be searched to determine presence of comp in TB
23     component_search("uvm_test")
24     component_search("uvm_env")
25     component_search("uvm_scoreboard")
26     component_search("uvm_agent")
27     component_search("uvm_driver")
28     component_search("uvm_monitor")
29     component_search("uvm_sequencer")
30     component_search("uvm_sequence ") # added space to accommodate full word matching
31     component_search("uvm_sequence_item")
32     f = open("tree_data.doc", "a+") #opens document for append
33     f.write("\nname list in table format\n")
34     f.write(str(x)) #draws updated table data
35     f.close()
36 def component_name_finder(keyword,line):
37     text = line.split() #to split line to get class name
38     name = text[1] #class name
39     print(keyword+"\t\t\t"+name+"\n")
40
41     print(keyword+"\t\t\t"+name+"\n")
42     f=open("tree_data.doc", "a+") #open and save in file tree data
43     f.write(keyword+"\t\t\t"+name+"\n")
44     x.add_row([keyword, name]) #adds new row to table
45     f.close()
46 tb_comps()

```

Results of component_name_finder.py script



```
component_name_finder x
D:\Python\Python392\python.exe "E:/uvm_tb_arch_doc_py-main/component_name_finder.py"
enter path of TB directory: E:\uvm_tb_arch_doc_py-main\dummy_tb
uvm_test          mem_model_test

uvm_env          mem_model_env

uvm_scoreboard    mem_scoreboard

uvm_agent         mem_agent_1

uvm_agent         mem_agent
|
uvm_driver        mem_driver_1

uvm_driver        mem_driver

uvm_monitor       mem_monitor_1

uvm_monitor       mem_monitor

uvm_sequencer     mem_sequencer_1

uvm_sequencer     mem_sequencer

uvm_sequence_item mem_seq_item
```

3.3 draw_rect_using_turtle.py

1)To draw the shapes using Turtle, the major shapes that are needed to construct the Testbench architecture are square and rectangle. Below is the Python script to draw square and rectangle and also connectivity

Steps:

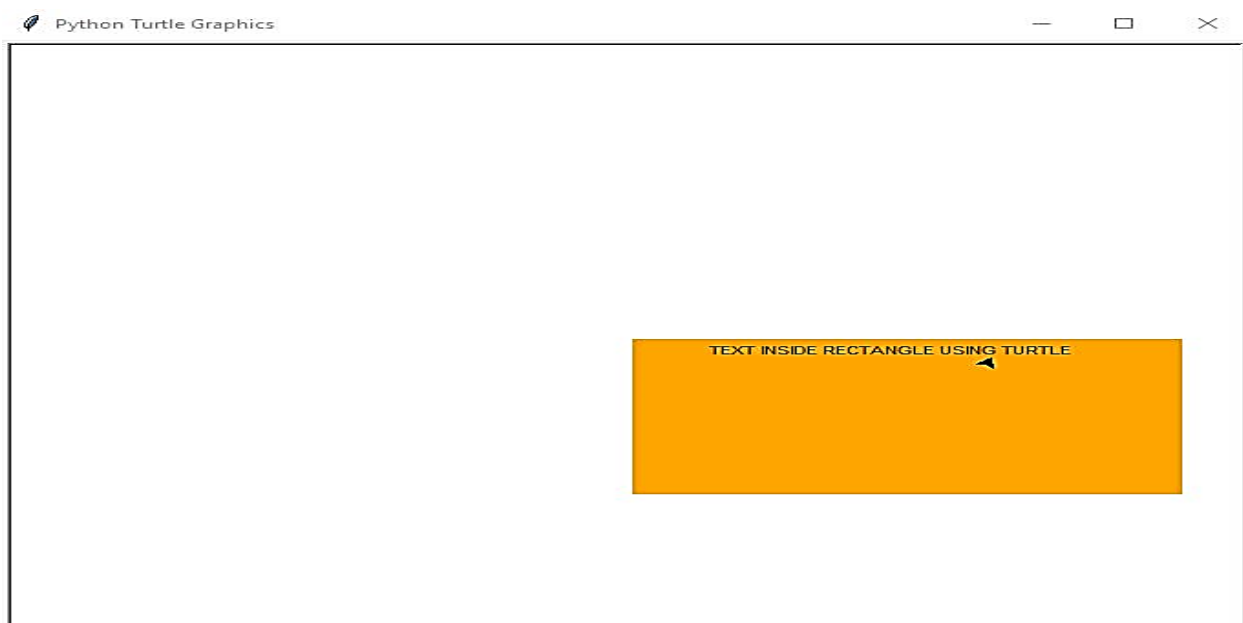
- 1.Import the turtle library
- 2.Set the Screen color
- 3.Instantiate the object for Turtle
- 4.Set the pen color
- 5.Define function draw_square and draw_rect. Pass the co-ordinates,length,width and color as the input to the function
- 6.Move to the desired location(co-ordinate) to draw the shape
- 7.Use begin_fill and end_fill to fill in the shape


```

1 #Choosing the TURTLE library
2 from turtle import *
3 # Choose Color for rectangle
4 color("orange")
5 # Enabling fill to color the shape
6 begin_fill()
7 # Traverse in directions, to draw rectangle
8 #Move forward direction of 300 units (length of rectangle)
9 forward(300);
10 #Move right direction of 90 units (For starting the breadth of rectangle)
11 right(90)
12 forward(150)
13 right(90)
14 forward(300);
15 right(90)
16 forward(150)
17 right(90)
18 # End the coloring inside that rectangle
19 end_fill()
20 #Choose color to write inside rectangle
21 color("BLACK") # Choose Black color to write
22 #Enabling the text fill color
23 begin_fill()
24 #This penup feature is to enable the pointer
25 penup()
26 #Fixing the pointer location from where to start the text inside rectangle
27 forward (150)
28 #right (45)
29 left(65)
30 backward (20)
31 #Write the desired text that needs to be written onto the rectangle
32 write("TEXT INSIDE RECTANGLE USING TURTLE", True, align="center")

```

Results of draw_rect_using_turtle.py script



3.4 draw_rect_with_text_inside.py(using Image draw module)

To draw the shapes using Image draw module, the major shapes that are needed to construct the Testbench architecture are square and rectangle. Below is the Python script to draw square and rectangle and also connectivity

Steps:

- Created an empty image *.jpg file
- Drew a rectangle in that image using ImageDraw module
- Saved the image & using the Image Draw module, inserted the text inside the rectangle (by changing the required dimensions in trial and error manner)
- Added that .jpg file into a .docx document & saved that. (using docx module)

```

1 import docx
2 from PIL import Image, ImageDraw
3 doc = docx.Document()
4 img = Image.new("RGB", (500, 500), "white")
5 # create a image draw handle
6 img1 = ImageDraw.Draw(img)
7 img1.rectangle((200, 125, 300, 200), fill = "orange", outline = "black", width = 1)
8 img1.text((210, 150), "CHECK TEXT", fill = "black", align = "center")
9 img.show()
10 img.save('E:\Python_task\\line.jpg');
11 doc.add_picture('E:\Python_task\\line.jpg');
12 doc.save('E:\Python_task\\pattern_printing_ex.docx');
..

```

Results of draw_rect_with_text_inside.py script(using Image draw module)



3.6 tb_arch_img_draw.py

To read the file which contains the information about Testbench and to draw the agent components

Steps:

1. Read the file tb_info.txt which contains the agent information (Number of agent). Read the first line and then using split method the words are stored in a list form which the number can be retrieved.
2. Define the method top, test, env, agent, scoreboard, driver, sequencer and monitor to draw the corresponding component
3. In the method definition, use the in-built function called rectangle from the ImageDraw module to draw the components
4. The arguments to the functions are:
 - x0,y0 – Starting co-ordinate to draw the rectangle
 - x1,y1 - End co-ordinate
 - Outline – Outline color for the shape
 - 1.Co-ordinates – where the text needs to be inserted
 - 2.Text in the form of string
 - 3.Color for the text
5. .Agents and its components are constructed based on the agent numbers. From env() the scoreboard() and agent components are called
6. The text method in Image Draw module issued to include the text. The parameters are: Coordinates, text, colour
7. The information collected from the testbench are written in a file. Here the tb_info.txt file contains agent information (number of agents)

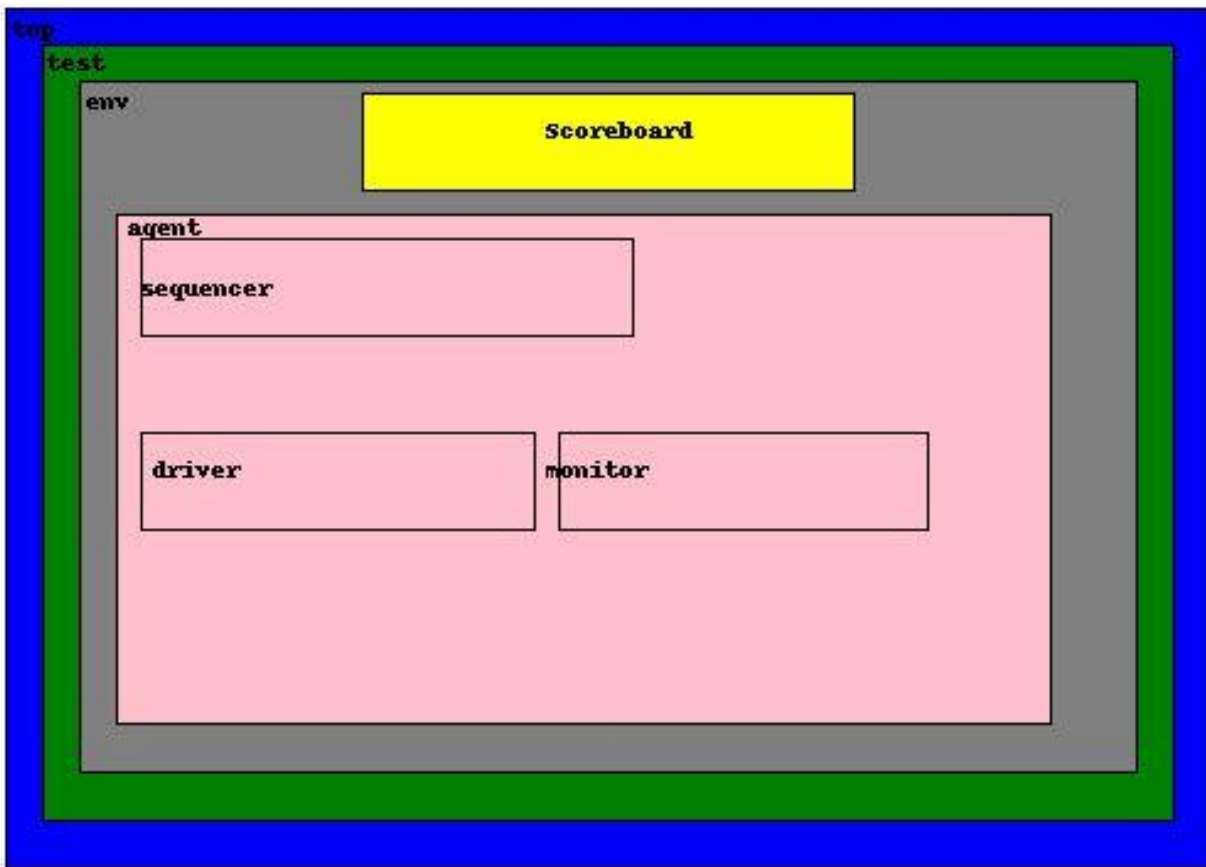
```

1 # Refer tb_arch_img_draw.docx for detailed explanation|
2 from PIL import ImageDraw,Image
3 f=open("tree_data.txt",'r')
4 content = f.readline()
5 agent=content.split()
6 agent_number=1;
7 f.close()
8 print(agent_number)
9 img=Image.new("RGB",(500,500),"white")
10 draw=ImageDraw.Draw(img)
11 def top():
12     draw.rectangle((5,5,495,360),fill="blue",outline="black")
13     draw.text((8,8),"top",fill="black")
14 def test():
15     draw.rectangle((20,20,480,340),fill="green",outline="black")
16     draw.text((22,22),"test",fill="black")
17 def env():
18     draw.rectangle((35,35,465,320),fill="grey",outline="black")
19     draw.text((38,38),"env",fill="black")
20 scoreboard()
21 for i in range(int(agent_number)):
22     agent(i)
23     sequencer(i)
24     driver(i)
25     monitor(i)
26 def scoreboard():
27     draw.rectangle((150,40,350,80),fill="yellow",outline="black")
28     draw.text((225,50),"Scoreboard",fill="black")
29 def agent(y):
30     x = y * ((380/int(agent_number)) + 10)
31     z=380/int(agent_number)
32     draw.rectangle((50+x,90,z+50+x,300),fill="pink",outline="black")
33     draw.text((55+x,90),"agent",fill="black")
34 def sequencer(y):
35     x = y * ((380/int(agent_number)) + 10)
36     z=200/int(agent_number)
37     draw.rectangle((60+x,100,z+60+x,140),fill="pink",outline="black")
38     draw.text((60+x,115),"sequencer",fill="black")
39 def driver(y):
40     x = y * ((380/int(agent_number)) + 10)
41     z=160/int(agent_number)
42     draw.rectangle((60+x,180,z+60+x,220),fill="pink",outline="black")
43     draw.text((65+x,190),"driver",fill="black")
44 def monitor(y):
45     x = y * ((380/int(agent_number)) + 10)
46     z=160/int(agent_number)
47     draw.rectangle((60+x+z+10,180,((z*2)+60+x),220),fill="pink",outline="black")
48     draw.text((65+x+z,190),"monitor",fill="black")
49 top()
50 test()
51 env()
52 img.show()

```

Results of tb_arch_img_draw.py script

```
tb_arch_img_draw x
D:\Python\Python392\python.exe E:/uvm_tb_arch_doc_py-main/tb_arch_img_draw.py
1
|
Process finished with exit code 0
>>
```



3.7 uvm_tb_arch_agent.py

- To draw TB Diagram aimed at 2 agentsTo draw Rectangle with given co-ordinate & fill with given colour and write the text inside rectangle
- To draw top, test, env blocks write the value of n choosen height of env block should be less , so we have to give proper dimensions
- This “docx” module is to manipulate with docs like MS Word. Used it to add TB diagram to the document, we have take handle doc for docx
- After we have to import tkinter & setting height & width to measure whole screen size and then
 - Create 1st outer rectangle for top
 - Create 2nd inner rectangle for test
 - Create 3rd inner rectangle for env
 - Create 5th inner rectangle for scoreboard
 - Create 4th inner rectangle for sequences, DUT, Interface, Virtual Interface
 - Check for No of agents user have to give
 - Start another rectangle MON inside agent
 - Start another rectangle DRV inside agent
 - Start another rectangle SEQR inside agent
- Then draw the arrows between Driver to Virtual Interface, Virtual Interface to Interface and Driver1 to Virtual Interface and then set the start & end co-ordinates
- By using arrowed line() method set the colour & thickness and then add the picture & save the picture in docx

```

1 #####METHOD TO DRAW RECTANGLE WITH THE GIVEN CO-ORDINATES AND FILL WITH THE GIVEN COLOR#####
2 def draw_rect(image,coordinates,fill,color,width=1):
3     rect_start = (coordinates[0][0],coordinates[0][1]);
4     rect_end = (coordinates[1][0], coordinates [1][1])
5     image.rectangle((rect_start,rect_end),fill=fill,outline = color)
6 #Method to write the text inside the rectangle
7 def wr_text_in_rect(image,start_wr_w,start_wr_h,str,tfill):
8     font = ImageFont.truetype("arial.ttf", 15)
9     image.text((start_wr_w,start_wr_h),str, fill = tfill,font = font)
10 #Method to draw the top,test,env blocks (w.r.t. the value of n chosen)
11 def call_simple_rect(w,h,n,text,bfill,tfill,img1):
12     w1 = w - (n*10); #end of x should be max
13     if n != 5:
14         h2 = h - (n*10); #end of 'y' should be max
15         h1 = n*15 + 10;
16         w2 = h1;
17     elif n == 5: #The height of the env block should be less; So used like below dimensions
18         h2 = h - (n*10*5)
19         h1 = n*15 + 10 + 35;
20         w2 = n*15 + 10;
21     top_right = (w1,h1)
22     bottom_left = (w2,h2)
23     start_x = w1 - (50);
24     start_y = h1 + (n*2);
25     outline_width = 10
26     outline_color = "black"
27     draw_rect(img1,(top_right, bottom_left), fill=bfill ,color=outline_color, width=outline_width)
28     wr_text_in_rect(img1,start_x,start_y,text,tfill)
29     print ("Dimensions are %0d %0d %0d %0d",top_right, bottom_left)
30     return w1;
31 #This docx module is to manipulate with docs like MS Word. Used it to add TB diagram to the document
32 import docx
33 #This opencv module in python ease us to draw arrowed line in the image
34 import cv2
35 # This pillow module to import Image draw module
36 from PIL import Image,ImageDraw,ImageFont
37 #Taking the handle doc for docx
38 doc = docx.Document()
39 #This Module is used to measure the whole screen size
40 import tkinter
41 root = tkinter.Tk()
42 width = root.winfo_screenwidth()
43 height = root.winfo_screenheight()
44 print ("Width & HEIGHT",width,height)
45 # create line image of width and height
46 w = width
47 h = height
48 img = Image.new("RGB", (w, h),"white")
49 img1 = ImageDraw.Draw(img)
50 #Create first outer rectangle top n=1
51 n = 1;
52 top_dim = call_simple_rect(w,h,n,"TOP","orange","black",img1);
53 #Create second inner rectangle test n=3
54 n = 3;
55 test_dim = call_simple_rect(w,h,n,"TEST","pink","black",img1);
56 #Create third inner rectangle env n=5
57 n = 5;
58 env_dim = call_simple_rect(w,h,n,"ENV","yellow","black",img1);
59 #Create fifth inner rectangle SCOREBOARD
60 top_right = (w-140,150)
61 bottom_left = (400,230)
62 start_x = ((w-140)+400)/2 + 12;
63 start_y = 180;
64 draw_rect(img1,(top_right, bottom_left), fill="gray" ,color="black", width=10)
65 wr_text_in_rect(img1,start_x,start_y,"SCOREBOARD","BLACK")
66 #Create fourth inner rectangle sequences DUT
67 top_right = (90,h-80)
68 bottom_left = (w-40,h-50)
69 start_x_dut = (90 + (w-40))/2;
70 start_y_dut = (((h - 80) + (h - 50))/2) - 12;
71 draw_rect(img1,(top_right, bottom_left), fill="grey" ,color="black", width=10)
72 wr_text_in_rect(img1,start_x_dut,start_y_dut,"DUT","BLACK")
73 #Create fourth inner rectangle Interface
74 top_right = (90,h-120)
75 bottom_left = (w-40,h-150)
76 start_x_if = (90 + (w-40))/2;
77 start_y_if = ((h-120)+(h-150))/2 - 12;
78 draw_rect(img1,(top_right, bottom_left), fill="green" ,color="black", width=10)

```



```

79 wr_text_in_rect(img1,start_x_if,start_y_if,"INTERFACE","BLACK")
80 #Create fourth inner rectangle UIF
81 top_right = (90,h-190)
82 bottom_left = (w-40,h-220)
83 start_x_vif = (90 + (w-40))/2;
84 start_y_vif = ((h-190)+(h-220))/2 - 12;
85 draw_rect(img1,(top_right, bottom_left), fill="gray",color="black", width=10)
86 wr_text_in_rect(img1,start_x_vif,start_y_vif,"UIF","BLACK")
87 print(env_dim);
88 #Check for number of agents
89 n = 7;
90 agnt_cnt = int(input("Enter no. of agents"))
91 w1 = (env_dim/agnt_cnt)
92 h1 = (env_dim/agnt_cnt)
93 tx = 40;
94 x0 = 0;
95 diff = 0;
96 m1 = 1;
97 m2 = 0;
98 #To draw the number of agents w.r.t. agent count
99 for val in range(agnt_cnt):
100     print("VALUE OF X0 IS",x0)
101     x1 = tx + 55;
102     print("VALUE OF X1 IS",x1)
103     y0 = (n*4*10)
104     if agnt_cnt == 1:
105         x0 = (env_dim/agnt_cnt) - 20;
106     elif agnt_cnt != 1:
107         x0 = (m1*(env_dim/agnt_cnt))+(m2*(x1 + diff));
108     print("VALUE OF X0 LATER IS",x0)
109     y1 = h - (4*n*10)
110     tx = x0;
111     diff = x0 - x1;
112     xdiff = x0 - x1;
113     ydiff = y1 - y0;
114     top_right = (x0,y0);
115     bottom_left = (x1,y1);
116     start_x = x0 - (19*xdiff/20);
117     start_y = y0 + (ydiff/20);
118
119     outline_width = 10
120     print(top_right);
121     print(bottom_left);
122     draw_rect(img1,(top_right, bottom_left), fill="cyan",color="black", width=outline_width)
123     wr_text_in_rect(img1,start_x,start_y,"AGENT","BLACK")
124     #Start another rectangle MONITOR inside the agent
125
126     x3 = x1 + (xdiff/20);
127     y3 = y1 - (ydiff/20);
128     x2 = x0 - (12*xdiff/20);
129     y2 = y0 + (12*ydiff/20);
130     top_right = (x2,y2);
131     bottom_left = (x3,y3);
132     xdiff_mon = x2 - x3;
133     ydiff_mon = y3 - y2;
134     start_x_mon = x3 + xdiff_mon/20;
135     start_y_mon = y2 + ydiff_mon/2;
136     outline_width = 10
137     draw_rect(img1,(top_right, bottom_left), fill="orange",color="black", width=outline_width)
138     #wr_text_in_rect(img1,start_x_mon,start_y_mon,"MONITOR","BLACK")
139     wr_text_in_rect(img1,start_x_mon,start_y_mon,"MON","BLACK")
140     #Start another rectangle DRIVER inside the agent
141     x5 = x1 + (12*xdiff/20);
142     y5 = y1 - (ydiff/20);
143     x4 = x0 - (xdiff/20);
144     y4 = y0 + (12*ydiff/20);
145     top_right = (x4,y4)
146
147
148     bottom_left = (x5,y5);
149     xdiff_drv = x4 - x5;
150     ydiff_drv = y5 - y4;
151     start_x_drv = x5 + xdiff_drv/20;
152     start_y_drv = y4 + ydiff_drv/2;
153     outline_width = 10
154     draw_rect(img1,(top_right, bottom_left), fill="green",color="black", width=outline_width)
155     wr_text_in_rect(img1,start_x_drv,start_y_drv,"DRV","BLACK")
156     m2 = 1;

```

```

157     m1 = 0;
158     print("DRIVER",start_x_dru,start_y_dru)
159     #Start another rectangle SEQUENCER inside the agent
160     x5 = x1 + (12*xdiff/20);
161     y5 = y1 - (12*ydiff/20);
162     x4 = x0 - (xdiff/20);
163     y4 = y0 + (ydiff/20);
164     top_right = (x4,y4);
165     bottom_left = (x5,y5);
166     xdiff_sqr = x4 - x5;
167     ydiff_sqr = y5 - y4;
168     start_x_sqr = x5 + xdiff_sqr / 20;
169     start_y_sqr = y4 + ydiff_sqr / 2;
170     outline_width = 10
171     draw_rect(img1,(top_right, bottom_left), fill="grey",color="black", width=outline_width)
172     wr_text_in_rect(img1,start_x_sqr,start_y_sqr,"SQR","BLACK")
173     m2 = 1;
174     m1 = 0;
175     img.show()
176     img.save('E:\Python_task\tb_arch.jpg');
177     # Arrow Drawing
178     path = 'E:\Python_task\tb_arch.jpg'
179     # Reading an image in default mode
180     image = cv2.imread(path)
181     # Window name in which image is displayed
182     window_name = 'Image'
183     #####DRAW ARROW BETWEEN DRIVER AND UIF#####
184     start_point = (int(start_x_dru - 15),int(start_y_dru) + 75)
185     # End coordinate
186     end_point = (int(start_x_dru - 15),int(start_y_dru + 25) + 115)
187     color = (0, 0, 0)
188     thickness = 3
189     # Using cv2.arrowedLine() method
190     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
191     cv2.imshow(window_name, image)
192     cv2.imwrite("E:\Python_task\tb_arch.jpg",image)
193     #####DRAW ARROW BETWEEN UIF AND INTERFACE#####
194     # Start coordinate
195     start_point = (int(start_x_vif - 15),int(start_y_vif + 25))
196
197     #####DRAW ARROW BETWEEN UIF AND INTERFACE#####
198     # Start coordinate
199     start_point = (int(start_x_vif - 15),int(start_y_vif + 25))
200     # End coordinate
201     end_point = (int(start_x_if - 15),int(start_y_if - 5))
202     color = (0, 0, 0)
203     thickness = 3
204     # Using cv2.arrowedLine() method
205     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
206     cv2.imshow(window_name, image)
207     #####DRAW ARROW BETWEEN INTERFACE AND DUT#####
208     start_point = (int(start_x_if - 15),int(start_y_if + 25))
209     # End coordinate
210     end_point = (int(start_x_dut - 15),int(start_y_dut - 5))
211     color = (0, 0, 0)
212     thickness = 3
213     # Using cv2.arrowedLine() method
214     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
215     cv2.imshow(window_name, image)
216     #####DRAW ARROW BETWEEN DRIVER1 AND UIF#####
217     start_point = (506,482)
218     # End coordinate
219     end_point = (506,547)
220     color = (0, 0, 0)
221     thickness = 3
222     # Using cv2.arrowedLine() method
223     image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
224     cv2.imshow(window_name, image)
225     cv2.imwrite("E:\Python_task\tb_arch.jpg",image)
226     doc.add_picture('E:\Python_task\tb_arch.jpg')
227     doc.save('E:\Python_task\pattern_printing_ex.docx')

```

Results of uvm_tb_arch_agent.py

```
uvm_tb_arch_agent
D:\Python\Python392\python.exe E:/uvm_tb_arch_doc_py-main/uvm_tb_arch_agent.py
Width & HEIGHT 1366 768
Dimensions are %0d %0d %0d %0d (1356, 25) (25, 758)
Dimensions are %0d %0d %0d %0d (1336, 55) (55, 738)
Dimensions are %0d %0d %0d %0d (1316, 120) (85, 518)
1316
Enter no. of agents 8
VALUE OF X0 IS 0
VALUE OF X1 IS 95
VALUE OF X0 LATER IS 164.5
(164.5, 280)
(95, 488)
DRIVER 137.91625 441.20000000000005
VALUE OF X0 IS 164.5
VALUE OF X1 IS 219.5
VALUE OF X0 LATER IS 289.0
(289.0, 280)
(219.5, 488)
DRIVER 262.41625 441.20000000000005
VALUE OF X0 IS 289.0
VALUE OF X1 IS 344.0
VALUE OF X0 LATER IS 413.5
(413.5, 280)
(344.0, 488)
DRIVER 386.91625 441.20000000000005
VALUE OF X0 IS 413.5
VALUE OF X1 IS 488.0
VALUE OF X0 LATER IS 518.0
(518.0, 280)
(488.0, 488)
DRIVER 511.41625 441.20000000000005
VALUE OF X0 IS 518.0
VALUE OF X1 IS 563.0
VALUE OF X0 LATER IS 637.5
(637.5, 280)
(563.0, 488)
DRIVER 635.91625 441.20000000000005
VALUE OF X0 IS 637.5
VALUE OF X1 IS 688.0
VALUE OF X0 LATER IS 768.0
(768.0, 280)
(688.0, 488)
DRIVER 760.41625 441.20000000000005
VALUE OF X0 IS 768.0
VALUE OF X1 IS 733.0
VALUE OF X0 LATER IS 858.0
(858.0, 280)
(733.0, 488)
DRIVER 884.91625 441.20000000000005
VALUE OF X0 IS 858.0
VALUE OF X1 IS 788.0
VALUE OF X0 LATER IS 958.0
(958.0, 280)
(788.0, 488)
DRIVER 1009.41625 441.20000000000005
VALUE OF X0 IS 958.0
VALUE OF X1 IS 833.0
VALUE OF X0 LATER IS 1048.0
(1048.0, 280)
(833.0, 488)
DRIVER 1133.91625 441.20000000000005
VALUE OF X0 IS 1048.0
VALUE OF X1 IS 878.0
VALUE OF X0 LATER IS 1138.0
(1138.0, 280)
(878.0, 488)
DRIVER 1258.41625 441.20000000000005
VALUE OF X0 IS 1138.0
VALUE OF X1 IS 923.0
VALUE OF X0 LATER IS 1228.0
(1228.0, 280)
(923.0, 488)
DRIVER 1382.91625 441.20000000000005
VALUE OF X0 IS 1228.0
VALUE OF X1 IS 968.0
VALUE OF X0 LATER IS 1318.0
(1318.0, 280)
(968.0, 488)
DRIVER 1507.41625 441.20000000000005
VALUE OF X0 IS 1318.0
VALUE OF X1 IS 1013.0
VALUE OF X0 LATER IS 1408.0
(1408.0, 280)
(1013.0, 488)
DRIVER 1631.91625 441.20000000000005
VALUE OF X0 IS 1408.0
VALUE OF X1 IS 1058.0
VALUE OF X0 LATER IS 1498.0
(1498.0, 280)
(1058.0, 488)
DRIVER 1756.41625 441.20000000000005
VALUE OF X0 IS 1498.0
VALUE OF X1 IS 1103.0
VALUE OF X0 LATER IS 1588.0
(1588.0, 280)
(1103.0, 488)
DRIVER 1880.91625 441.20000000000005
VALUE OF X0 IS 1588.0
VALUE OF X1 IS 1148.0
VALUE OF X0 LATER IS 1678.0
(1678.0, 280)
(1148.0, 488)
DRIVER 2005.41625 441.20000000000005
VALUE OF X0 IS 1678.0
VALUE OF X1 IS 1193.0
VALUE OF X0 LATER IS 1768.0
(1768.0, 280)
(1193.0, 488)
DRIVER 2129.91625 441.20000000000005
VALUE OF X0 IS 1768.0
VALUE OF X1 IS 1238.0
VALUE OF X0 LATER IS 1858.0
(1858.0, 280)
(1238.0, 488)
DRIVER 2254.41625 441.20000000000005
VALUE OF X0 IS 1858.0
VALUE OF X1 IS 1283.0
VALUE OF X0 LATER IS 1948.0
(1948.0, 280)
(1283.0, 488)
DRIVER 2378.91625 441.20000000000005
VALUE OF X0 IS 1948.0
VALUE OF X1 IS 1328.0
VALUE OF X0 LATER IS 2038.0
(2038.0, 280)
(1328.0, 488)
DRIVER 2503.41625 441.20000000000005
VALUE OF X0 IS 2038.0
VALUE OF X1 IS 1373.0
VALUE OF X0 LATER IS 2128.0
(2128.0, 280)
(1373.0, 488)
DRIVER 2627.91625 441.20000000000005
VALUE OF X0 IS 2128.0
VALUE OF X1 IS 1418.0
VALUE OF X0 LATER IS 2218.0
(2218.0, 280)
(1418.0, 488)
DRIVER 2752.41625 441.20000000000005
VALUE OF X0 IS 2218.0
VALUE OF X1 IS 1463.0
VALUE OF X0 LATER IS 2308.0
(2308.0, 280)
(1463.0, 488)
DRIVER 2876.91625 441.20000000000005
VALUE OF X0 IS 2308.0
VALUE OF X1 IS 1508.0
VALUE OF X0 LATER IS 2398.0
(2398.0, 280)
(1508.0, 488)
DRIVER 3001.41625 441.20000000000005
VALUE OF X0 IS 2398.0
VALUE OF X1 IS 1553.0
VALUE OF X0 LATER IS 2488.0
(2488.0, 280)
(1553.0, 488)
DRIVER 3125.91625 441.20000000000005
VALUE OF X0 IS 2488.0
VALUE OF X1 IS 1598.0
VALUE OF X0 LATER IS 2578.0
(2578.0, 280)
(1598.0, 488)
DRIVER 3250.41625 441.20000000000005
VALUE OF X0 IS 2578.0
VALUE OF X1 IS 1643.0
VALUE OF X0 LATER IS 2668.0
(2668.0, 280)
(1643.0, 488)
DRIVER 3374.91625 441.20000000000005
VALUE OF X0 IS 2668.0
VALUE OF X1 IS 1688.0
VALUE OF X0 LATER IS 2758.0
(2758.0, 280)
(1688.0, 488)
DRIVER 3500.41625 441.20000000000005
VALUE OF X0 IS 2758.0
VALUE OF X1 IS 1733.0
VALUE OF X0 LATER IS 2848.0
(2848.0, 280)
(1733.0, 488)
DRIVER 3624.91625 441.20000000000005
VALUE OF X0 IS 2848.0
VALUE OF X1 IS 1778.0
VALUE OF X0 LATER IS 2938.0
(2938.0, 280)
(1778.0, 488)
DRIVER 3749.41625 441.20000000000005
VALUE OF X0 IS 2938.0
VALUE OF X1 IS 1823.0
VALUE OF X0 LATER IS 3028.0
(3028.0, 280)
(1823.0, 488)
DRIVER 3873.91625 441.20000000000005
VALUE OF X0 IS 3028.0
VALUE OF X1 IS 1868.0
VALUE OF X0 LATER IS 3118.0
(3118.0, 280)
(1868.0, 488)
DRIVER 4000.41625 441.20000000000005
VALUE OF X0 IS 3118.0
VALUE OF X1 IS 1913.0
VALUE OF X0 LATER IS 3208.0
(3208.0, 280)
(1913.0, 488)
DRIVER 4124.91625 441.20000000000005
VALUE OF X0 IS 3208.0
VALUE OF X1 IS 1958.0
VALUE OF X0 LATER IS 3298.0
(3298.0, 280)
(1958.0, 488)
DRIVER 4249.41625 441.20000000000005
VALUE OF X0 IS 3298.0
VALUE OF X1 IS 2003.0
VALUE OF X0 LATER IS 3388.0
(3388.0, 280)
(2003.0, 488)
DRIVER 4373.91625 441.20000000000005
VALUE OF X0 IS 3388.0
VALUE OF X1 IS 2048.0
VALUE OF X0 LATER IS 3478.0
(3478.0, 280)
(2048.0, 488)
DRIVER 4500.41625 441.20000000000005
VALUE OF X0 IS 3478.0
VALUE OF X1 IS 2093.0
VALUE OF X0 LATER IS 3568.0
(3568.0, 280)
(2093.0, 488)
DRIVER 4624.91625 441.20000000000005
VALUE OF X0 IS 3568.0
VALUE OF X1 IS 2138.0
VALUE OF X0 LATER IS 3658.0
(3658.0, 280)
(2138.0, 488)
DRIVER 4749.41625 441.20000000000005
VALUE OF X0 IS 3658.0
VALUE OF X1 IS 2183.0
VALUE OF X0 LATER IS 3748.0
(3748.0, 280)
(2183.0, 488)
DRIVER 4873.91625 441.20000000000005
VALUE OF X0 IS 3748.0
VALUE OF X1 IS 2228.0
VALUE OF X0 LATER IS 3838.0
(3838.0, 280)
(2228.0, 488)
DRIVER 5000.41625 441.20000000000005
VALUE OF X0 IS 3838.0
VALUE OF X1 IS 2273.0
VALUE OF X0 LATER IS 3928.0
(3928.0, 280)
(2273.0, 488)
DRIVER 5124.91625 441.20000000000005
VALUE OF X0 IS 3928.0
VALUE OF X1 IS 2318.0
VALUE OF X0 LATER IS 4018.0
(4018.0, 280)
(2318.0, 488)
DRIVER 5249.41625 441.20000000000005
VALUE OF X0 IS 4018.0
VALUE OF X1 IS 2363.0
VALUE OF X0 LATER IS 4108.0
(4108.0, 280)
(2363.0, 488)
DRIVER 5373.91625 441.20000000000005
VALUE OF X0 IS 4108.0
VALUE OF X1 IS 2408.0
VALUE OF X0 LATER IS 4198.0
(4198.0, 280)
(2408.0, 488)
DRIVER 5500.41625 441.20000000000005
VALUE OF X0 IS 4198.0
VALUE OF X1 IS 2453.0
VALUE OF X0 LATER IS 4288.0
(4288.0, 280)
(2453.0, 488)
DRIVER 5624.91625 441.20000000000005
VALUE OF X0 IS 4288.0
VALUE OF X1 IS 2498.0
VALUE OF X0 LATER IS 4378.0
(4378.0, 280)
(2498.0, 488)
DRIVER 5749.41625 441.20000000000005
VALUE OF X0 IS 4378.0
VALUE OF X1 IS 2543.0
VALUE OF X0 LATER IS 4468.0
(4468.0, 280)
(2543.0, 488)
DRIVER 5873.91625 441.20000000000005
VALUE OF X0 IS 4468.0
VALUE OF X1 IS 2588.0
VALUE OF X0 LATER IS 4558.0
(4558.0, 280)
(2588.0, 488)
DRIVER 6000.41625 441.20000000000005
VALUE OF X0 IS 4558.0
VALUE OF X1 IS 2633.0
VALUE OF X0 LATER IS 4648.0
(4648.0, 280)
(2633.0, 488)
DRIVER 6124.91625 441.20000000000005
VALUE OF X0 IS 4648.0
VALUE OF X1 IS 2678.0
VALUE OF X0 LATER IS 4738.0
(4738.0, 280)
(2678.0, 488)
DRIVER 6249.41625 441.20000000000005
VALUE OF X0 IS 4738.0
VALUE OF X1 IS 2723.0
VALUE OF X0 LATER IS 4828.0
(4828.0, 280)
(2723.0, 488)
DRIVER 6373.91625 441.20000000000005
VALUE OF X0 IS 4828.0
VALUE OF X1 IS 2768.0
VALUE OF X0 LATER IS 4918.0
(4918.0, 280)
(2768.0, 488)
DRIVER 6500.41625 441.20000000000005
VALUE OF X0 IS 4918.0
VALUE OF X1 IS 2813.0
VALUE OF X0 LATER IS 5008.0
(5008.0, 280)
(2813.0, 488)
DRIVER 6624.91625 441.20000000000005
VALUE OF X0 IS 5008.0
VALUE OF X1 IS 2858.0
VALUE OF X0 LATER IS 5098.0
(5098.0, 280)
(2858.0, 488)
DRIVER 6749.41625 441.20000000000005
VALUE OF X0 IS 5098.0
VALUE OF X1 IS 2903.0
VALUE OF X0 LATER IS 5188.0
(5188.0, 280)
(2903.0, 488)
DRIVER 6873.91625 441.20000000000005
VALUE OF X0 IS 5188.0
VALUE OF X1 IS 2948.0
VALUE OF X0 LATER IS 5278.0
(5278.0, 280)
(2948.0, 488)
DRIVER 7000.41625 441.20000000000005
VALUE OF X0 IS 5278.0
VALUE OF X1 IS 2993.0
VALUE OF X0 LATER IS 5368.0
(5368.0, 280)
(2993.0, 488)
DRIVER 7124.91625 441.20000000000005
VALUE OF X0 IS 5368.0
VALUE OF X1 IS 3038.0
VALUE OF X0 LATER IS 5458.0
(5458.0, 280)
(3038.0, 488)
DRIVER 7249.41625 441.20000000000005
VALUE OF X0 IS 5458.0
VALUE OF X1 IS 3083.0
VALUE OF X0 LATER IS 5548.0
(5548.0, 280)
(3083.0, 488)
DRIVER 7373.91625 441.20000000000005
VALUE OF X0 IS 5548.0
VALUE OF X1 IS 3128.0
VALUE OF X0 LATER IS 5638.0
(5638.0, 280)
(3128.0, 488)
DRIVER 7500.41625 441.20000000000005
VALUE OF X0 IS 5638.0
VALUE OF X1 IS 3173.0
VALUE OF X0 LATER IS 5728.0
(5728.0, 280)
(3173.0, 488)
DRIVER 7624.91625 441.20000000000005
VALUE OF X0 IS 5728.0
VALUE OF X1 IS 3218.0
VALUE OF X0 LATER IS 5818.0
(5818.0, 280)
(3218.0, 488)
DRIVER 7749.41625 441.20000000000005
VALUE OF X0 IS 5818.0
VALUE OF X1 IS 3263.0
VALUE OF X0 LATER IS 5908.0
(5908.0, 280)
(3263.0, 488)
DRIVER 7873.91625 441.20000000000005
VALUE OF X0 IS 5908.0
VALUE OF X1 IS 3308.0
VALUE OF X0 LATER IS 5998.0
(5998.0, 280)
(3308.0, 488)
DRIVER 8000.41625 441.20000000000005
VALUE OF X0 IS 5998.0
VALUE OF X1 IS 3353.0
VALUE OF X0 LATER IS 6088.0
(6088.0, 280)
(3353.0, 488)
DRIVER 8124.91625 441.20000000000005
VALUE OF X0 IS 6088.0
VALUE OF X1 IS 3398.0
VALUE OF X0 LATER IS 6178.0
(6178.0, 280)
(3398.0, 488)
DRIVER 8249.41625 441.20000000000005
VALUE OF X0 IS 6178.0
VALUE OF X1 IS 3443.0
VALUE OF X0 LATER IS 6268.0
(6268.0, 280)
(3443.0, 488)
DRIVER 8373.91625 441.20000000000005
VALUE OF X0 IS 6268.0
VALUE OF X1 IS 3488.0
VALUE OF X0 LATER IS 6358.0
(6358.0, 280)
(3488.0, 488)
DRIVER 8500.41625 441.20000000000005
VALUE OF X0 IS 6358.0
VALUE OF X1 IS 3533.0
VALUE OF X0 LATER IS 6448.0
(6448.0, 280)
(3533.0, 488)
DRIVER 8624.91625 441.20000000000005
VALUE OF X0 IS 6448.0
VALUE OF X1 IS 3578.0
VALUE OF X0 LATER IS 6538.0
(6538.0, 280)
(3578.0, 488)
DRIVER 8749.41625 441.20000000000005
VALUE OF X0 IS 6538.0
VALUE OF X1 IS 3623.0
VALUE OF X0 LATER IS 6628.0
(6628.0, 280)
(3623.0, 488)
DRIVER 8873.91625 441.20000000000005
VALUE OF X0 IS 6628.0
VALUE OF X1 IS 3668.0
VALUE OF X0 LATER IS 6718.0
(6718.0, 280)
(3668.0, 488)
DRIVER 9000.41625 441.20000000000005
VALUE OF X0 IS 6718.0
VALUE OF X1 IS 3713.0
VALUE OF X0 LATER IS 6808.0
(6808.0, 280)
(3713.0, 488)
DRIVER 9124.91625 441.20000000000005
VALUE OF X0 IS 6808.0
VALUE OF X1 IS 3758.0
VALUE OF X0 LATER IS 6898.0
(6898.0, 280)
(3758.0, 488)
DRIVER 9249.41625 441.20000000000005
VALUE OF X0 IS 6898.0
VALUE OF X1 IS 3803.0
VALUE OF X0 LATER IS 6988.0
(6988.0, 280)
(3803.0, 488)
DRIVER 9373.91625 441.20000000000005
VALUE OF X0 IS 6988.0
VALUE OF X1 IS 3848.0
VALUE OF X0 LATER IS 7078.0
(7078.0, 280)
(3848.0, 488)
DRIVER 9500.41625 441.20000000000005
VALUE OF X0 IS 7078.0
VALUE OF X1 IS 3893.0
VALUE OF X0 LATER IS 7168.0
(7168.0, 280)
(3893.0, 488)
DRIVER 9624.91625 441.20000000000005
VALUE OF X0 IS 7168.0
VALUE OF X1 IS 3938.0
VALUE OF X0 LATER IS 7258.0
(7258.0, 280)
(3938.0, 488)
DRIVER 9749.41625 441.20000000000005
VALUE OF X0 IS 7258.0
VALUE OF X1 IS 3983.0
VALUE OF X0 LATER IS 7348.0
(7348.0, 280)
(3983.0, 488)
DRIVER 9873.91625 441.20000000000005
VALUE OF X0 IS 7348.0
VALUE OF X1 IS 4028.0
VALUE OF X0 LATER IS 7438.0
(7438.0, 280)
(4028.0, 488)
DRIVER 10000.41625 441.20000000000005
VALUE OF X0 IS 7438.0
VALUE OF X1 IS 4073.0
VALUE OF X0 LATER IS 7528.0
(7528.0, 280)
(4073.0, 488)
DRIVER 10124.91625 441.20000000000005
VALUE OF X0 IS 7528.0
VALUE OF X1 IS 4118.0
VALUE OF X0 LATER IS 7618.0
(7618.0, 280)
(4118.0, 488)
DRIVER 10249.41625 441.20000000000005
VALUE OF X0 IS 7618.0
VALUE OF X1 IS 4163.0
VALUE OF X0 LATER IS 7708.0
(7708.0, 280)
(4163.0, 488)
DRIVER 10373.91625 441.20000000000005
VALUE OF X0 IS 7708.0
VALUE OF X1 IS 4208.0
VALUE OF X0 LATER IS 7798.0
(7798.0, 280)
(4208.0, 488)
DRIVER 10500.41625 441.20000000000005
VALUE OF X0 IS 7798.0
VALUE OF X1 IS 4253.0
VALUE OF X0 LATER IS 7888.0
(7888.0, 280)
(4253.0, 488)
DRIVER 10624.91625 441.20000000000005
VALUE OF X0 IS 7888.0
VALUE OF X1 IS 4298.0
VALUE OF X0 LATER IS 7978.0
(7978.0, 280)
(4298.0, 488)
DRIVER 10749.41625 441.20000000000005
VALUE OF X0 IS 7978.0
VALUE OF X1 IS 4343.0
VALUE OF X0 LATER IS 8068.0
(8068.0, 280)
(4343.0, 488)
DRIVER 10873.91625 441.20000000000005
VALUE OF X0 IS 8068.0
VALUE OF X1 IS 4388.0
VALUE OF X0 LATER IS 8158.0
(8158.0, 280)
(4388.0, 488)
DRIVER 11000.41625 441.20000000000005
VALUE OF X0 IS 8158.0
VALUE OF X1 IS 4433.0
VALUE OF X0 LATER IS 8248.0
(8248.0, 280)
(4433.0, 488)
DRIVER 11124.91625 441.20000000000005
VALUE OF X0 IS 8248.0
VALUE OF X1 IS 4478.0
VALUE OF X0 LATER IS 8338.0
(8338.0, 280)
(4478.0, 488)
DRIVER 11249.41625 441.20000000000005
VALUE OF X0 IS 8338.0
VALUE OF X1 IS 4523.0
VALUE OF X0 LATER IS 8428.0
(8428.0, 280)
(4523.0, 488)
DRIVER 11373.91625 441.20000000000005
VALUE OF X0 IS 8428.0
VALUE OF X1 IS 4568.0
VALUE OF X0 LATER IS 8518.0
(8518.0, 280)
(4568.0, 488)
DRIVER 11500.41625 441.20000000000005
VALUE OF X0 IS 8518.0
VALUE OF X1 IS 4613.0
VALUE OF X0 LATER IS 8608.0
(8608.0, 280)
(4613.0, 488)
DRIVER 11624.91625 441.20000000000005
VALUE OF X0 IS 8608.0
VALUE OF X1 IS 4658.0
VALUE OF X0 LATER IS 8698.0
(8698.0, 280)
(4658.0, 488)
DRIVER 11749.41625 441.20000000000005
VALUE OF X0 IS 8698.0
VALUE OF X1 IS 4703.0
VALUE OF X0 LATER IS 8788.0
(8788.0, 280)
(4703.0, 488)
DRIVER 11873.91625 441.20000000000005
VALUE OF X0 IS 8788.0
VALUE OF X1 IS 4748.0
VALUE OF X0 LATER IS 8878.0
(8878.0, 280)
(4748.0, 488)
DRIVER 12000.41625 441.20000000000005
VALUE OF X0 IS 8878.0
VALUE OF X1 IS 4793.0
VALUE OF X0 LATER IS 8968.0
(8968.0, 280)
(4793.0, 488)
DRIVER 12124.91625 441.20000000000005
VALUE OF X0 IS 8968.0
VALUE OF X1 IS 4838.0
VALUE OF X0 LATER IS 9058.0
(9058.0, 280)
(4838.0, 488)
DRIVER 12249.41625 441.20000000000005
VALUE OF X0 IS 9058.0
VALUE OF X1 IS 4883.0
VALUE OF X0 LATER IS 9148.0
(9148.0, 280)
(4883.0, 488)
DRIVER 12373.91625 441.20000000000005
VALUE OF X0 IS 9148.0
VALUE OF X1 IS 4928.0
VALUE OF X0 LATER IS 9238.0
(9238.0, 280)
(4928.0, 488)
DRIVER 12500.41625 441.20000000000005
VALUE OF X0 IS 9238.0
VALUE OF X1 IS 4973.0
VALUE OF X0 LATER IS 9328.0
(9328.0, 280)
(4973.0, 488)
DRIVER 12624.91625 441.20000000000005
VALUE OF X0 IS 9328.0
VALUE OF X1 IS 5018.0
VALUE OF X0 LATER IS 9418.0
(9418.0, 280)
(5018.0, 488)
DRIVER 12749.41625 441.20000000000005
VALUE OF X0 IS 9418.0
VALUE OF X1 IS 5063.0
VALUE OF X0 LATER IS 9508.0
(9508.0, 280)
(5063.0, 488)
DRIVER 12873.91625 441.20000000000005
VALUE OF X0 IS 9508.0
VALUE OF X1 IS 5108.0
VALUE OF X0 LATER IS 9598.0
(9598.0, 280)
(5108.0, 488)
DRIVER 13000.41625 441.20000000000005
VALUE OF X0 IS 9598.0
VALUE OF X1 IS 5153.0
VALUE OF X0 LATER IS 9688.0
(9688.0, 280)
(5153.0, 488)
DRIVER 13124.91625 441.20000000000005
VALUE OF X0 IS 9688.0
VALUE OF X1 IS 5198.0
VALUE OF X0 LATER IS 9778.0
(9778.0, 280)
(5198.0, 488)
DRIVER 13249.41625 441.20000000000005
VALUE OF X0 IS 9778.0
VALUE OF X1 IS 5243.0
VALUE OF X0 LATER IS 9868.0
(9868.0, 280)
(5243.0, 488)
DRIVER 13373.91625 441.20000000000005
VALUE OF X0 IS 9868.0
VALUE OF X1 IS 5288.0
VALUE OF X0 LATER IS 9958.0
(9958.0, 280)
(5288.0, 488)
DRIVER 13500.41625 441.20000000000005
VALUE OF X0 IS 9958.0
VALUE OF X1 IS 5333.0
VALUE OF X0 LATER IS 10048.0
(10048.0, 280)
(5333.0, 488)
DRIVER 13624.91625 441.20000000000005
VALUE OF X0 IS 10048.0
VALUE OF X1 IS 5378.0
VALUE OF X0 LATER IS 10138.0
(10138.0, 280)
(5378.0, 488)
DRIVER 13749.41625 441.20000000000005
VALUE OF X0 IS 10138.0
VALUE OF X1 IS 5423.0
VALUE OF X0 LATER IS 10228.0
(10228.0, 280)
(5423.0, 488)
DRIVER 13873.91625 441.20000000000005
VALUE OF X0 IS 10228.0
VALUE OF X1 IS 5468.0
VALUE OF X0 LATER IS 10318.0
(10318.0, 280)
(5468.0, 488)
DRIVER 14000.41625 441.20000000000005
VALUE OF X0 IS 10318.0
VALUE OF X1 IS 5513.0
VALUE OF X0 LATER IS 10408.0
(10408.0, 280)
(5513.0, 488)
DRIVER 14124.91625 441.20000000000005
VALUE OF X0 IS 10408.0
VALUE OF X1 IS 5558.0
VALUE OF X0 LATER IS 10498.0
(10498.0, 280)
(5558.0, 488)
DRIVER 14249.41625 441.20000000000005
VALUE OF X0 IS 10498.0
VALUE OF X1 IS 5603.0
VALUE OF X0 LATER IS 10588.0
(10588.0, 280)
(5603.0, 488)
DRIVER 14373.91625 441.20000000000005
VALUE OF X0 IS 10588.0
VALUE OF X1 IS 5648.0
VALUE OF X0 LATER IS 10678.0
(10678.0, 280)
(5648.0, 488)
DRIVER 14500.41625 441.20000000000005
VALUE OF X0 IS 10678.0
VALUE OF X1 IS 5693.0
VALUE OF X0 LATER IS 10768.0
(10768.0, 280)
(5693.0, 488)
DRIVER 14624.91625 441.20000000000005
VALUE OF X0 IS 10768.0
VALUE OF X1 IS 5738.0
VALUE OF X0 LATER IS 10858.0
(10858.0, 280)
(5738.0, 488)
DRIVER 14749.41625 441.20000000000005
VALUE OF X0 IS 10858.0
VALUE OF X1 IS 5783.0
VALUE OF X0 LATER IS 10948.0
(10948.0, 280)
(5783.0, 488)
DRIVER 14873.91625 441.20000000000005
VALUE OF
```

3.8 uvm_tb_name_image_file_generator.py

- Firstly import the PIL(Python Imaging Library) which is Pillow, it adds image processing capabilities to python interpreter.
- This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. Import tkinter package (tk interface) is a standard python interface to the tk GUI toolkit.
- Now adding pillow and tkinter for image draw and canvas. The Canvas is a rectangular area intended for drawing pictures or other complex layouts. Import image , imagedraw and imagefont from pillow.
- Then import OS library which is the module in python provides functions for interacting with the operating system.this module provides a portable way of using operating system-dependent functionality.
- Now user enters the path of TB directory. Set width and height according to the screen size. Set canvas size, create lookup table for uvm component and name. Create file for writing component and component name, also clean file before writing the data.
- Now drawing top level structure, set the co-ordinates and set color for top as yellow color. Drawing test level structure, set the co-ordinates and set color for top as orange color. Drawing env level structure, set the co-ordinates and set color for top as yellow color.
- Drawing scoreboard level structure, set the co-ordinates and set color for top as red. Drawing agent level structure, set the co-ordinates and set color for top as blue color also set the image font.
- Call the rectangular creation function. So we are doing here, user enter the input as TB directory and it will create block diagram of the component like- top, test, env, agent, scoreboard

```

1 #####METHOD TO DRAW RECTANGLE WITH THE GIVEN CO-ORDINATES AND FILL WITH THE GIVEN COLOR#####
2 path = input("enter path of TB directory: ") #user input for path of TB directory
3
4 import tkinter
5 import os,re #search library
6 root = tkinter.Tk()
7 lookup = {"keyword":"name"} #lookup table created for uvm component and name
8 f = open("tree_data.txt","w") #create file for writing component and component name
9 def component_search(keyword): # to search for keyword in all files of directory
10     root_dir = path
11     for root, dirs, files in os.walk(root_dir, onerror=None, topdown=True): # to loop inside all files of directory
12         for filename in files:
13             file_path = os.path.join(root, filename)
14             with open(file_path, "rb") as f: # read file as binary
15                 for line in f:
16                     line = line.decode("utf-8") #decode to string for read
17                     if keyword in line: # keyword determines the word to be looked into in each of the file
18                         component_name_finder(keyword, line) # call function to find class name
19
20 def read_file_draw():
21     f = open("tree_data.txt","r") #read file written with names
22     file = f.read()
23     global a
24     #print (file)
25     if (re.search("uvm_test",file)): # find keyword in file
26         #top() # draw top
27         #test(lookup['uvm_test']) #draw test with found name
28         #env(lookup['uvm_env']) # draw env with found name
29         #scoreboard(lookup['uvm_scoreboard']) # draw scb with found name
30         a = file.count("uvm_agent") # find number of agents
31         print(a)
32         print ("number of agents : "+str(a)) #print number of agents
33         # a=3 '''uncomment and add values for agents explicitly'''
34         #for i in range(a):
35             #agent(a,i,lookup['uvm_agent']) # draw agent with agent names
36     f.close()
37 def tb_comps(): #generic word in tb components that can be searched to determine presence of comp in TB
38     component_search("uvm_test")
39     component_search("uvm_env")
40
41     component_search("uvm_scoreboard")
42     component_search("uvm_agent")
43     component_search("uvm_driver")
44     component_search("uvm_monitor")
45     component_search("uvm_sequencer")
46     component_search("uvm_sequence ")
47     component_search("uvm_sequence_item")
48     component_search("interface")
49     #component_search("package")
50
51     for key, value in lookup.items(): #saving component names into lookup table
52         print(key, ' : ', value)
53 def component_name_finder(keyword,line):
54     text = line.split() #split line where text found
55     name = text[1] # get name from split line
56     #print(keyword,name)
57     lookup [keyword] = name #add name to lookup table
58     f = open("tree_data.txt","a+") #add found name to file : open write and close
59     f.write(keyword+"\t\t\t\t"+name+"\n")
60     f.close()
61
62 tb_comps()
63 read_file_draw()
64 font_value=input("enter a font_value:")
65 font_size=int(font_value)
66 print("value of font size",font_size);
67 def draw_rect(image,coordinates,fill,color,width=1):
68     rect_start = (coordinates[0][0],coordinates[0][1]);
69     rect_end = (coordinates[1][0], coordinates [1][1]);
70     image.rectangle((rect_start,rect_end),fill=fill,outline = color)
71 #Method to write the text inside the rectangle
72 def wr_text_in_rect(image,start_wr_w,start_wr_h,str,tfill):
73     font = ImageFont.truetype("arial.ttf",font_size)
74     image.text((start_wr_w,start_wr_h),str, fill = tfill,font = font)
75 #Method to draw the top,test,env blocks (w.r.t. the value of n chosen)
76 def call_simple_rect(w,h,n,text,bfill,tfill,img1):
77     w1 = w - (n*10); #end of x should be max
78     if n != 5:
79         h2 = h - (n*10); #end of 'y' should be max

```



```

79         h1 = n*15 + 10;
80         w2 = h1;
81         elif n == 5: #The height of the env block should be less; So used like below dimensions
82             h2 = h - (n*10*5)
83             h1 = n*15 + 10 + 35;
84             w2 = n*15 + 10;
85         top_right = (w1,h1)
86         bottom_left = (w2,h2)
87         start_x = w1 - (50);
88         start_y = h1 + (n*2);
89         outline_width = 10
90         outline_color = "black"
91         draw_rect(img1,(top_right, bottom_left), fill=bfill ,color=outline_color, width=outline_width)
92         wr_text_in_rect(img1,start_x,start_y,text,tfill)
93         print ("Dimensions are %0d %0d %0d %0d",top_right, bottom_left)
94         return w1;
95 #This docx module is to manipulate with docs like MS Word. Used it to add TB diagram to the document
96 import docx
97
98 #This opencv module in python ease us to draw arrowed line in the image
99 import cv2
100 # This pillow module to import Image draw module
101 from PIL import Image,ImageDraw,ImageFont
102 #Taking the handle doc for docx
103 doc = docx.Document()
104 #This Module is used to measure the whole screen size
105
106 #root = tkinter.Tk()
107 #lookup = {"keyword":"name"} #lookup table created for uvm component and name
108
109 width = root.winfo_screenwidth()
110 height = root.winfo_screenheight()
111 print ("Width & HEIGHT",width,height)
112 # create line image of width and height
113 w = width
114 h = height
115 img = Image.new("RGB", (w, h),"white")
116 img1 = ImageDraw.Draw(img)
117 #Create first outer rectangle top n=1
118
119 #def top():
120 n = 1;
121 top_dim = call_simple_rect(w,h,n,"TOP","white","black",img1);#orange
122 #Create second inner rectangle test n=3
123 #def test(name):
124 n = 3;
125 #label = name
126 test_dim = call_simple_rect(w,h,n,"TEST","white","black",img1);#pink
127 #Create third inner rectangle env n=5
128 n = 5;
129 env_dim = call_simple_rect(w,h,n,"ENV","white","black",img1);#yellow
130 #Create fifth inner rectangle SCOREBOARD
131 top_right = (w-140,150)
132 bottom_left = (400,230)
133 start_x = ((w-140)+400)/2 + 12;
134 start_y = 180;
135 draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=10)#gray
136 wr_text_in_rect(img1,start_x,start_y,"SCOREBOARD","BLACK")
137 #Create fourth inner rectangle sequences DUT
138 top_right = (90,h-80)
139 bottom_left = (w-40,h-50)
140 start_x_dut = (90 + (w-40))/2;
141 start_y_dut = (((h - 80) + (h - 50))/2) - 12;
142 draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=10)#grey
143 wr_text_in_rect(img1,start_x_dut,start_y_dut,"DUT","BLACK")
144 #Create fourth inner rectangle Interface
145 top_right = (90,h-120)
146 bottom_left = (w-40,h-150)
147 start_x_if = (90 + (w-40))/2;
148 start_y_if = ((h-120)+(h-150))/2 - 12;
149 draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=10)#green
150 wr_text_in_rect(img1,start_x_if,start_y_if,"INTERFACE","BLACK")
151 #Create fourth inner rectangle UIF
152 top_right = (90,h-190)
153 bottom_left = (w-40,h-220)
154 start_x_vif = (90 + (w-40))/2;
155 start_y_vif = ((h-190)+(h-220))/2 - 12;
156 draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=10)#gray
157 wr_text_in_rect(img1,start_x_vif,start_y_vif,"UIF","BLACK")

```

```

157 print(env_dim);
158 #Check for number of agents
159 n = 7;
160 agnt_cnt = 1 #int(input("Enter no. of agents"))
161 print("Navn_ag_cnt")
162 print(agnt_cnt)
163 w1 = (env_dim/agnt_cnt)
164 h1 = (env_dim/agnt_cnt)
165 tx = 40;
166 x0 = 0;
167 diff = 0;
168 m1 = 1;
169 m2 = 0;
170 #To draw the number of agents w.r.t. agent count
171 for val in range(agnt_cnt):
172     print("VALUE OF X0 IS",x0)
173     x1 = tx + 55;
174     print("VALUE OF X1 IS",x1)
175     y0 = (n*4*10)
176     if agnt_cnt == 1:
177         x0 = (env_dim/agnt_cnt) - 20;
178         print("VALUE OF X0 IS IF AGENT==1",x0)
179     elif agnt_cnt != 1:
180         x0 = (m1*(env_dim/agnt_cnt))+(m2*(x1 + diff));
181     print("VALUE OF X0 LATER IS",x0)
182     y1 = h - (4*n*10)
183     tx = x0;
184     diff = x0 - x1;
185     xdiff = x0 - x1;
186     ydiff = y1 - y0;
187     top_right = (x0,y0);
188     bottom_left = (x1,y1);
189     #start_x = x0 - 50;
190     start_x = x0 - (19*xdiff/20);
191     #start_y = y0 + 5;
192     start_y = y0 + (ydiff/20);
193
194     outline_width = 10
195     print(top_right);

196     print(bottom_left);
197     draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=outline_width)#cyan
198     wr_text_in_rect(img1,start_x,start_y,"AGENT","BLACK")
199     #Start another rectangle MONITOR inside the agent
200
201     x3 = x1 + (xdiff/20);
202     #y3 = y0 + 20
203     y3 = y1 - (ydiff/20);
204     #x2 = x0 - 350
205     x2 = x0 - (12*xdiff/20);
206     #y2 = y0 + 100
207     y2 = y0 + (12*ydiff/20);
208     top_right = (x2,y2);
209     bottom_left = (x3,y3);
210     xdiff_mon = x2 - x3;
211     ydiff_mon = y3 - y2;
212     #start_x_mon = (x2 + x3)/2;
213     start_x_mon = x3 + xdiff_mon/20;
214     #start_y_mon = y3 + 15;
215     start_y_mon = y2 + ydiff_mon/2;
216     outline_width = 10
217     draw_rect(img1,(top_right, bottom_left), fill="white" ,color="black", width=outline_width)#orange
218     #wr_text_in_rect(img1,start_x_mon,start_y_mon,"MONITOR","BLACK")
219     wr_text_in_rect(img1,start_x_mon,start_y_mon,"MON","BLACK")
220     #Start another rectangle DRIVER inside the agent
221     # x5 = x3 + 300
222     x5 = x1 + (12*xdiff/20);
223     # y5 = y3 + 100
224     y5 = y1 - (ydiff/20);
225     #x4 = x2 + 300
226     x4 = x0 - (xdiff/20);
227     #y4 = y2 + 100
228     y4 = y0 + (12*ydiff/20);
229     top_right = (x4,y4);
230     bottom_left = (x5,y5);
231     xdiff_drv = x4 - x5;
232     ydiff_drv = y5 - y4;
233     #start_x_drv = (x4 + x5)/2;
234     start_y_drv = y5 + 5;

```



```

235 start_x_drv = x5 + xdiff_drv/20;
236 start_y_drv = y4 + ydiff_drv/2;
237 outline_width = 10
238 draw_rect(img1,(top_right, bottom_left), fill="white",color="black", width=outline_width)#green
239 #wr_text_in_rect(img1,start_x_drv,start_y_drv,"DRIVER","BLACK")
240 wr_text_in_rect(img1,start_x_drv,start_y_drv,"DRV","BLACK")
241 m2 = 1;
242 m1 = 0;
243 print("DRIVER",start_x_drv,start_y_drv);|
244 #Start another rectangle SEQUENCER inside the agen
245 #x5 = x3 + 300
246 x5 = x1 + (12*xdiff/20);
247 #y5 = y3
248 y5 = y1 - (12*ydiff/20);
249 #x4 = x2 + 300
250 x4 = x0 - (xdiff/20);
251 #y4 = y2
252 y4 = y0 + (ydiff/20);
253 top_right = (x4,y4);
254 bottom_left = (x5,y5);
255 xdiff_sqr = x4 - x5;
256 ydiff_sqr = y5 - y4;
257 #start_x_sqr = (x4 + x5)/2;
258 #start_y_sqr = y5 + 5;
259 start_x_sqr = x5 + xdiff_sqr / 20;
260 start_y_sqr = y4 + ydiff_sqr / 2;
261 outline_width = 10
262 draw_rect(img1,(top_right, bottom_left), fill="white",color="black", width=outline_width)#grey
263 #wr_text_in_rect(img1,start_x_sqr,start_y_sqr,"SEQUENCER","BLACK")
264 wr_text_in_rect(img1,start_x_sqr,start_y_sqr,"SQR","BLACK")
265 m2 = 1;
266 m1 = 0;
267 img.show()
268 img.save('E:\Python_task\tb_arch.jpg');
269 # Arrow Drawing
270 path = 'E:\Python_task\tb_arch.jpg'
271 # Reading an image in default mode
272 image = cv2.imread(path)
273 # Window name in which image is displayed
-----
274 window_name = 'Image'
275 #####DRAW ARROW BETWEEN DRIVER AND UIF#####
276 start_point = (int(start_x_drv - 25),int(start_y_drv) + 75)
277 # End coordinate
278 end_point = (int(start_x_drv - 25),int(start_y_drv + 25) + 115)
279 color = (0, 0, 0)
280 thickness = 3
281 # Using cv2.arrowedLine() method
282 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
283 cv2.imshow(window_name, image)
284 cv2.imwrite("E:\Python_task\tb_arch.jpg",image)
285 #####DRAW ARROW BETWEEN UIF AND INTERFACE#####
286 # Start coordinate
287 start_point = (int(start_x_vif - 15),int(start_y_vif + 25))
288 # End coordinate
289 end_point = (int(start_x_if - 15),int(start_y_if - 5))
290 color = (0, 0, 0)
291 thickness = 3
292 # Using cv2.arrowedLine() method
293 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
294 cv2.imshow(window_name, image)
295
296 #####DRAW ARROW BETWEEN INTERFACE AND DUT#####
297 start_point = (int(start_x_if - 15),int(start_y_if + 25))
298 # End coordinate
299 end_point = (int(start_x_dut - 15),int(start_y_dut - 5))
300 color = (0, 0, 0)
301 thickness = 3
302 # Using cv2.arrowedLine() method
303 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
304 cv2.imshow(window_name, image)
305 #tb_comps()
306 #read_file_draw()
307 #####DRAW ARROW BETWEEN DRIVER1 AND UIF#####
308 start_point = (506,482)
309 # End coordinate
310 end_point = (506,547)
311 color = (0, 0, 0)
312 thickness = 3

```



```

313 # Using cv2.arrowedLine() method
314 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
315 #tb_comps()
316 #read_file_draw()
317 cv2.imshow(window_name, image)
318 #####DRAW ARROW BETWEEN DRIVER2 AND VIF#####
319 start_point = (836,482)
320 # End coordinate
321 end_point = (836,547)
322 color = (0, 0, 0)
323 thickness = 3
324 # Using cv2.arrowedLine() method
325 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
326 #tb_comps()
327 #read_file_draw()
328 cv2.imshow(window_name, image)
329 #####DRAW ARROW BETWEEN DRIVER2 AND VIF#####
330 start_point = (236,482)
331 # End coordinate
332 end_point = (236,547)
333 color = (0, 0, 0)
334 thickness = 3
335 # Using cv2.arrowedLine() method
336 image = cv2.arrowedLine(image, start_point, end_point,color, thickness)
337 #tb_comps()
338 #read_file_draw()
339 cv2.imshow(window_name, image)
340 cv2.imwrite("E:\Python_task\tb_arch.jpg",image)
341 doc.add_picture('E:\Python_task\tb_arch.jpg')
342 doc.save('E:\Python_task\pattern_printing_ex.docx')
~

```

Results of Create uvm_tb_name_image_file_generator:

```

↑ enter path of TB directory: E:\uvm_tb_arch_doc_py-main\dummy_tb
↓ keyword : name
| uvm_test : mem_model_test
| uvm_env : mem_model_env
| uvm_scoreboard : mem_scoreboard
| uvm_agent : mem_agent
| uvm_driver : mem_driver
| uvm_monitor : mem_monitor
| uvm_sequencer : mem_sequencer
| uvm_sequence_item : mem_seq_item
| interface : instance,
| 2
| number of agents : 2
| enter a font_value:14
| value of font_sixe 14
| Width & HEIGHT 1366 768
| Dimensions are %0d %0d %0d %0d (1356, 25) (25, 758)
| Dimensions are %0d %0d %0d %0d (1336, 55) (55, 738)
| Dimensions are %0d %0d %0d %0d (1316, 120) (85, 518)
| 1316
| Navn_ag_cnt
| 1
| VALUE OF X0 IS 0
| VALUE OF X1 IS 95
| VALUE OF X0 IS IF AGENT==1 1296.0
| VALUE OF X0 LATER IS 1296.0

```

Results of Create uvm_tb_name_image_generator:

