DAILY ASSESSMENT FORMAT

Date:	**28-05-2020**	**Name:**	**BINDUSHRI**
Course:	**Logic designs**	**USN:**	**4AL17EC011**
Topic:	**1.boolean equation for digital circuits** **Cominational circuits:conversion of mux and decoders.** **2.mux to logic gates** **3.7 seg decoder with common anode display.**		**6th A**
Github Repository:	**Bindushri**		

FORENOON SESSION DETAILS

28-05-2020 Logic Design

] Boolean equation for digital circuits :: Combinational Circuits : conversion of NOR & Decoders to logic gates.

→ Boolean algebra is a system of mathematical logic
→ It is defined with : Set of elements, a set of operators, and a number of axioms or postulates
* Set of elements - (0, 1)
* three binary operators - OR * AND (+) * (·)

Axiom or postulates of Boolean algebra are a set of logical expression upon which we build a useful theorem.

"AND operation"	"OR operation"	"NOT operation"
→ 0.0 = 0	→ 0+0 = 0	→ $\overline{0}$ or 0' → 1
→ 0.1 = 0	→ 0+1 = 1	→ $\overline{1}$ or 1' → 0
→ 1.0 = 0	→ 1+0 = 1	
→ 1.1 = 1	→ 1+1 = 1	

In Boolean Algebra
A+A = A or 1+1 = 1 * A.A = A or 1.1 = 1

ordinary algebra.
A+A = 2A A.A = A^2.
1+1 = 2 1.1 = 1

In Binary numbers system
1+1 = 10 1.1 = 1

Postulate

x+0 → x	x.0 = 0	$\overline{(\overline{x})}$ or $(x')' = x$
x+1 → 1	x.1 = x	
$x+\overline{x} \rightarrow 1$	x.x = x	
x+x ≠ x	$x.\overline{x} = 0$	

Identity element :- ex :- x+0 = x (additive) now here 0 is identity because any number added with x then value should be x.
0+x = x

Multiplication → x.1 = x
↳ multiplication identity

1) Commutative law 2) Associative law 3) Distributive law

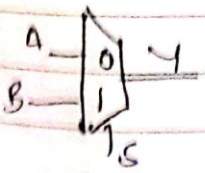
$x+y = y+x$ $\quad x+(y+z) = (x+y)+z$ $\quad x(y+z) = xy+xz$

Theorems of Boolean

1) Absorption theorem 2) $x+xy = x+xy = x$, 3) $x+\bar{x}y = x+y$

MUX to logic gates

1) Nand, Nor - universal gates
2) "Universal logic"
3) Mux and decoders are called universal logic
4) 2:1 mux can be used to create different logic gates



$2^n - 1$

2^n = inputs $\quad n$ = selection lines

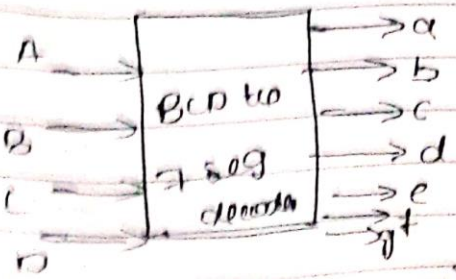
0	A
1	B

↓ S

A S' + B S

* Invert do

* Design of 7 segment decoder with common anode display.



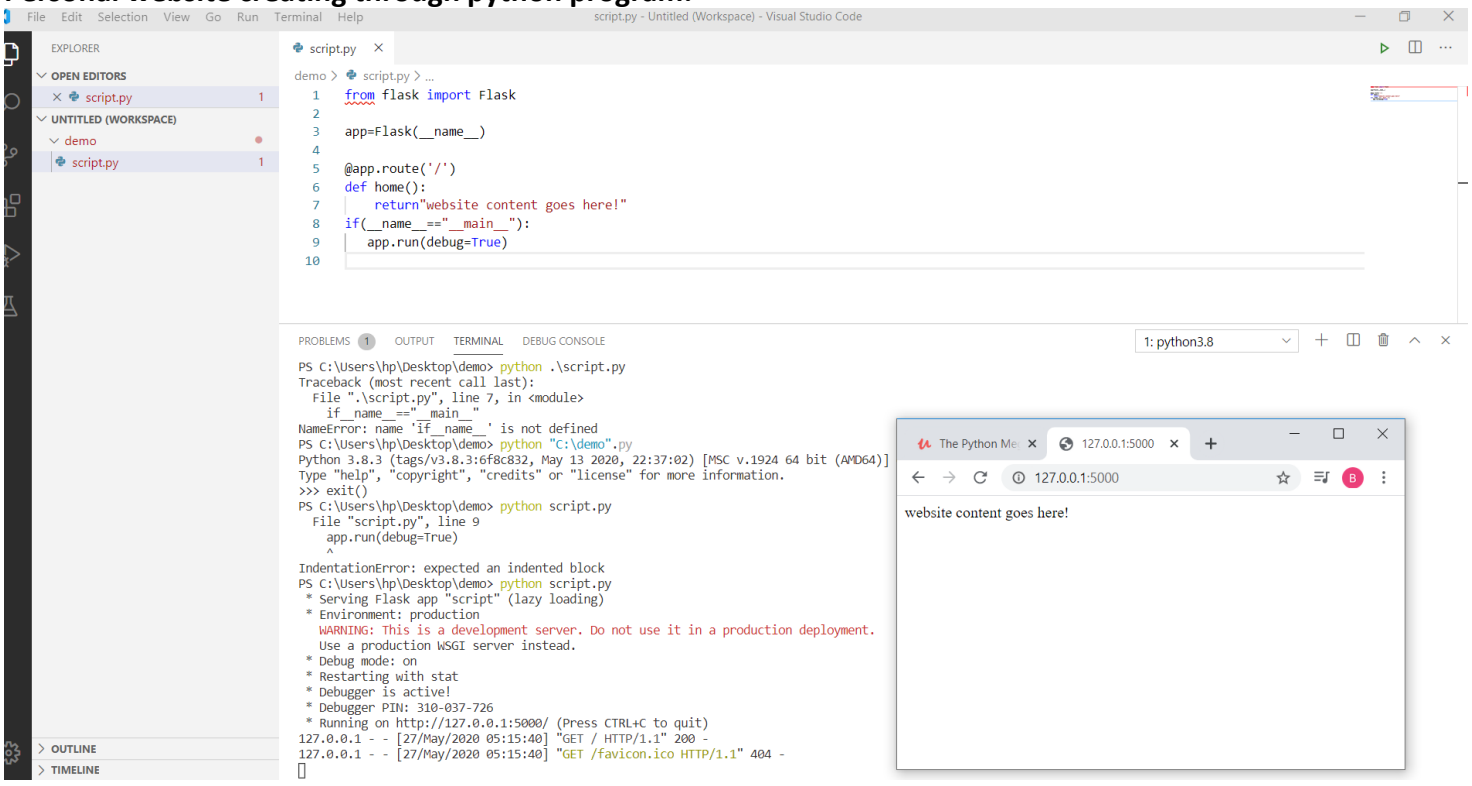
Date:28may2020
Course: python
Topic:
Basics:sec24

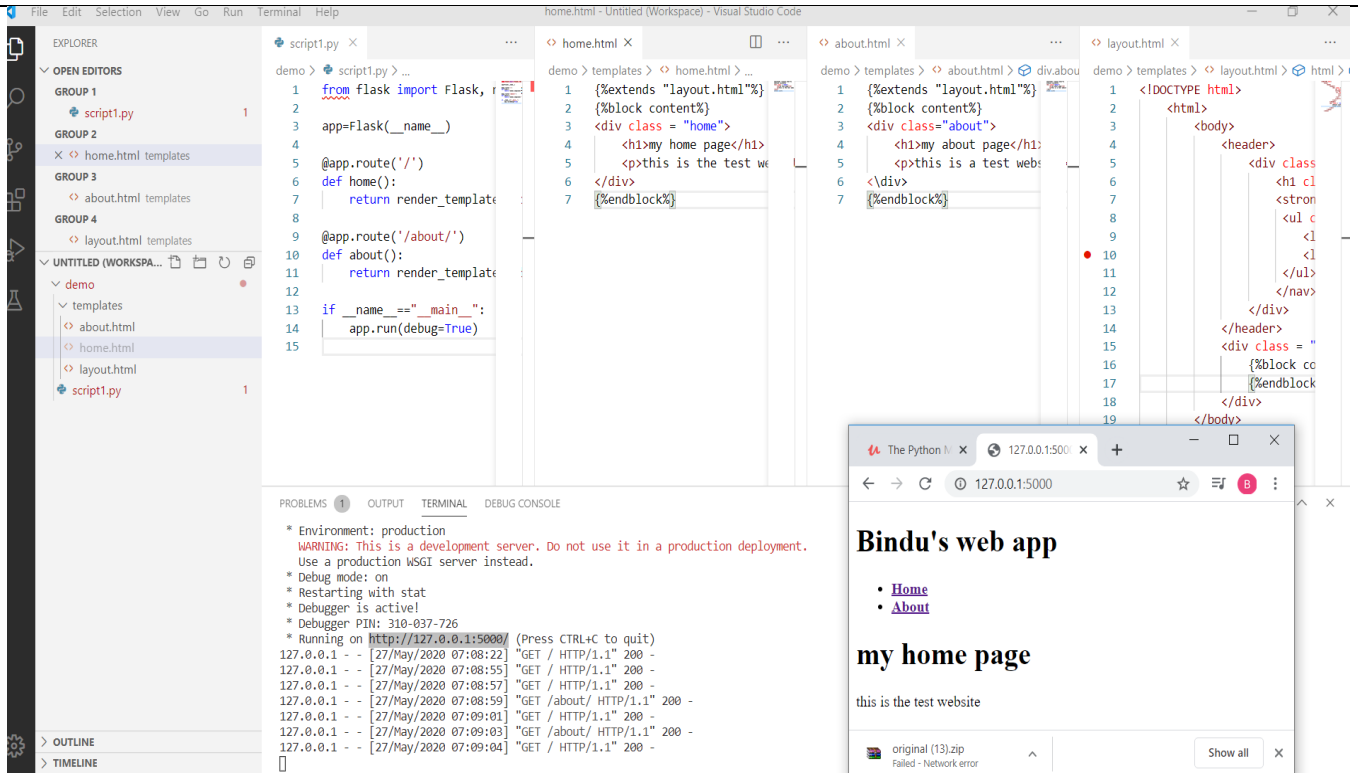
Name:Bindushri
USN:4AL17EC011
Sem&Sec:6th A

AFTERNOON SESSION DETAILS

Image of session

Personal website creating through python program:



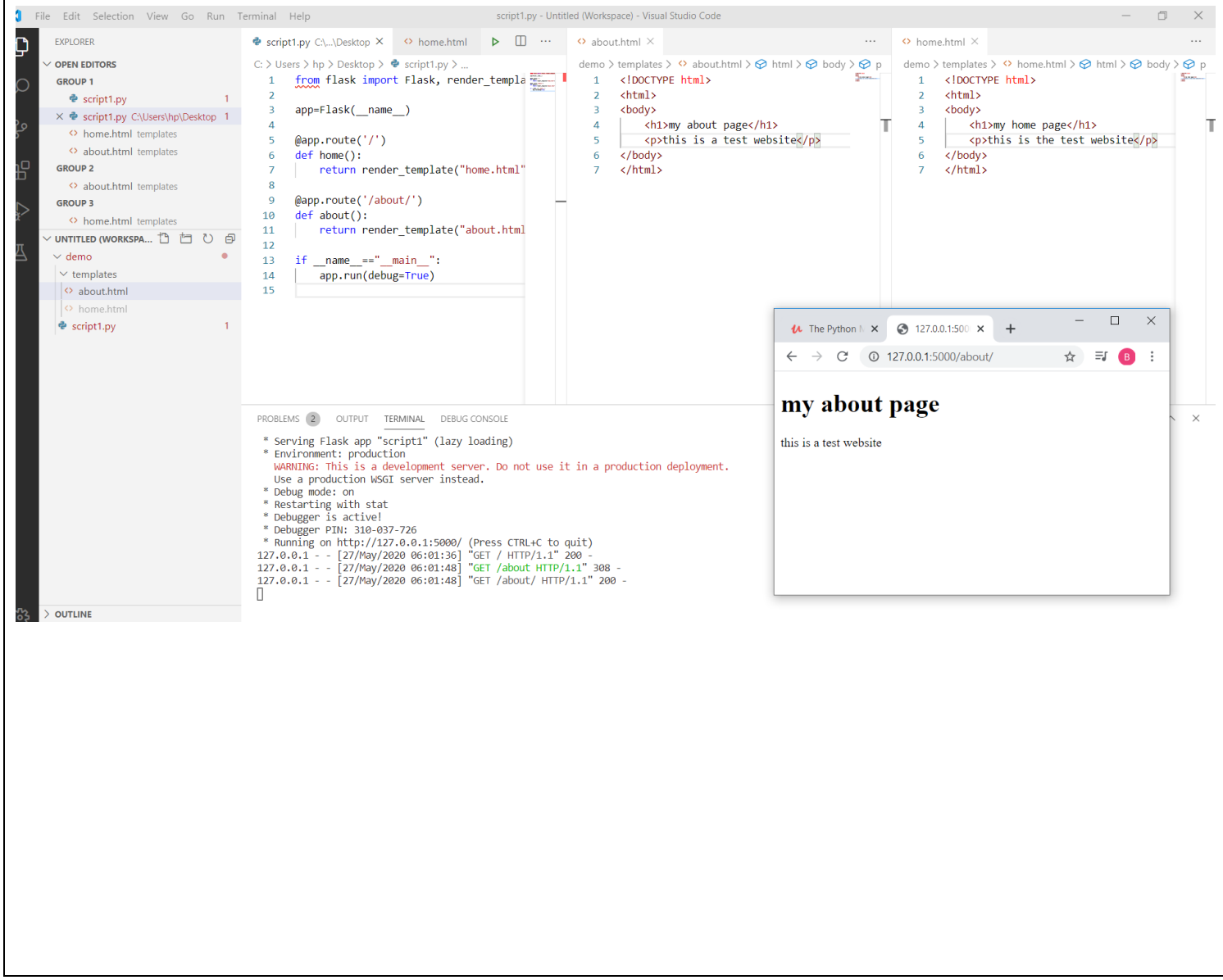

File Edit Selection View Go Run Terminal Help
home.html - Untitled (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
GROUP 1
script1.py
GROUP 2
home.html templates
GROUP 3
about.html templates
GROUP 4
layout.html templates
UNTITLED (WORKSPA...
demo
templates
about.html
home.html
layout.html
script1.py
OUTLINE
TIMELINE
script1.py
home.html
about.html
layout.html
from flask import Flask,
app=Flask(__name__)
@app.route('/')
def home():
return render_templat
@app.route('/about/')
def about():
return render_templat
if __name__=="__main__":
app.run(debug=True)
{%extends "layout.html"%}
{%block content%}

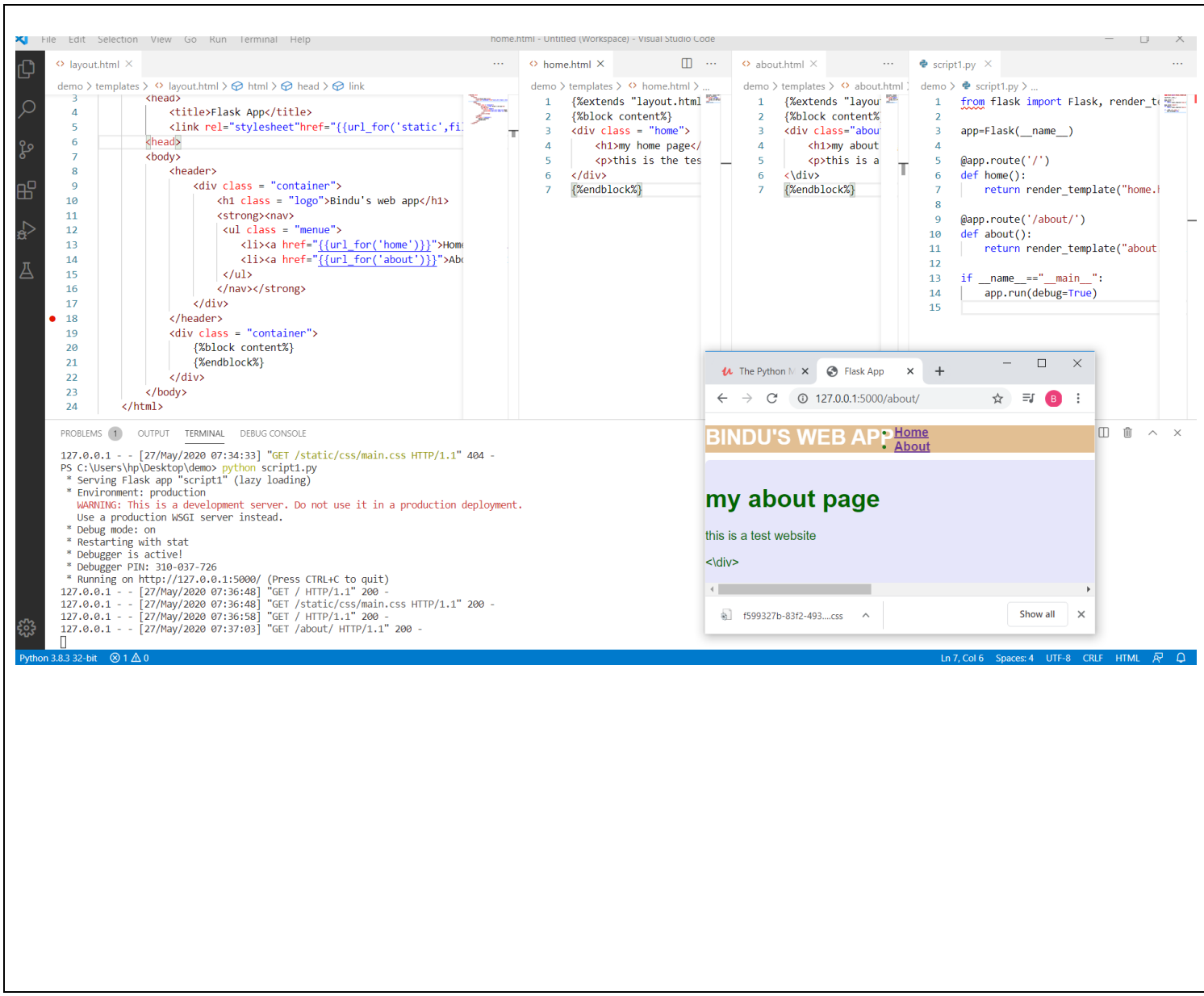
<h1>my home page</h1>
<p>this is the test we

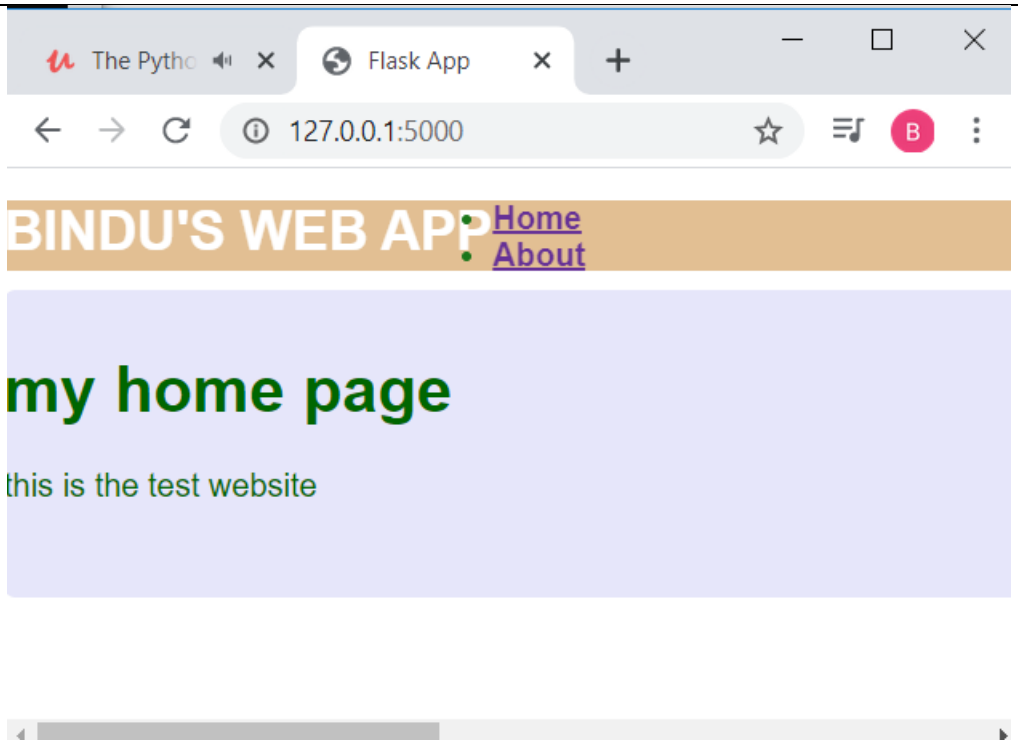
{%endblock%}
{%extends "layout.html"%}
{%block content%}

<h1>my about page</h1>
<p>this is a test webs

{%endblock%}
<!DOCTYPE html>
<html>
<body>
<header>
PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 310-037-726
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2020 07:08:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:08:55] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:08:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:08:59] "GET /about/ HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:09:03] "GET /about/ HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:09:04] "GET / HTTP/1.1" 200 -
127.0.0.1:5000
Bindu's web app
Home
About
my home page
this is the test website
original (13).zip
Failed - Network error
Show all


File Edit Selection View Go Run Terminal Help
script1.py - Untitled (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
GROUP 1
script1.py
script1.py C:\Users\hp\Desktop
home.html templates
about.html templates
GROUP 2
about.html templates
GROUP 3
home.html templates
UNTITLED (WORKSPA...
demo
templates
about.html
home.html
script1.py
OUTLINE
script1.py C:\...\Desktop
home.html
about.html
home.html
C: > Users > hp > Desktop > script1.py > ...
from flask import Flask, render_templa
app=Flask(__name__)
@app.route('/')
def home():
return render_template("home.html"
@app.route('/about/')
def about():
return render_template("about.html
if __name__=="__main__":
app.run(debug=True)
demo > templates > about.html > html > body > p
<!DOCTYPE html>
<html>
<body>
<h1>my about page</h1>
this is a test website
</body>
</html>
demo > templates > home.html > html > body > p
<!DOCTYPE html>
<html>
<body>
<h1>my home page</h1>
this is the test website
</body>
</html>
PROBLEMS 2
OUTPUT
TERMINAL
DEBUG CONSOLE
* Serving Flask app "script1" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 310-037-726
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2020 06:01:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 06:01:48] "GET /about HTTP/1.1" 308 -
127.0.0.1 - - [27/May/2020 06:01:48] "GET /about/ HTTP/1.1" 200 -
The Python
127.0.0.1:500
127.0.0.1:5000/about/
my about page
this is a test website


layout.html
home.html
about.html
script1.py
<head>
<title>Flask App</title>
<body>
<header>
<h1 class = "logo">Bindu's web app</h1>
<ul class = "menue">
<a href="{{url_for('home')}}">Home
<a href="{{url_for('about')}}">About
{%block content%}
{%endblock%}
<h1>my home page</
<p>this is the tes
<h1>my about
<p>this is a
from flask import Flask, render_t
app=Flask(__name__)
@app.route('/')
def home():
return render_template("home.
@app.route('/about/')
def about():
return render_template("about
if __name__=="__main__":
app.run(debug=True)
PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
127.0.0.1 - - [27/May/2020 07:34:33] "GET /static/css/main.css HTTP/1.1" 404 -
PS C:\Users\hp\Desktop\demo> python script1.py
* Serving Flask app "script1" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 310-037-726
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2020 07:36:48] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:36:48] "GET /static/css/main.css HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:36:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2020 07:37:03] "GET /about/ HTTP/1.1" 200 -
Flask App
127.0.0.1:5000/about/
BINDU'S WEB APP
Home
About
my about page
this is a test website
<\div>
f599327b-83f2-493....css
Show all
Python 3.8.3 32-bit
Ln 7, Col 6
Spaces: 4
UTF-8
CRLF
HTML



Script.py:

```
from flask import Flask, render_template

app=Flask(__name__)

@app.route('/')
def home():
    return render_template("home.html")

@app.route('/about/')
def about():
    return render_template("about.html")

if __name__=="__main__":
    app.run(debug=True)
```

home.html code:

```
{%extends "layout.html"%}
{%block content%}
<div class = "home">
    <h1>my home page</h1>
    <p>this is the test website</p>
</div>
{%endblock%}
```

About.html code:

```
{%extends "layout.html"%}
{%block content%}
<div class="about">
  <h1>my about page</h1>
  <p>this is a test website</p>
<\div>
{%endblock%}
```

Layout.html code:

```
<!DOCTYPE html>
  <html>
    <head>
      <title>Flask App</title>
      <link rel="stylesheet"href="{{url_for('static',filename='css/main.css')}}"
    <head>
    <body>
      <header>
        <div class = "container">
          <h1 class = "logo">Bindu's web app</h1>
          <strong><nav>
           <ul class = "menue">
             <li><a href="{{url_for('home')}}">Home</a></li>
             <li><a href="{{url_for('about')}}">About</a></li>
           </ul>
          </nav></strong>
        </div>
```

```
    </header>
    <div class = "container">
      {%block content%}
      {%endblock%}
    </div>
  </body>
</html>
```

28-05-2020 Lec:- 24 . Object oriented programming.

Program process same as Lec 23. But need to add some of lines.

* Turning this application into OOP style.

* Inserting the class in previous code.

* OOP is not only adding some function inside class. But it is also about good design.

* creating Bank account object

Demo → Caccount → ac.py & balance.txt

acc.py

```
1.  class Account:
2.
3.      def __init__(self, filepath):
4.          self.filepath = filepath
5.          with open(filepath, 'r') as file:
6.              self.balance = int(file.read())
7.
8.      def withdraw(self, amount):
9.          self.balance = self.balance - amount
10.
11.     def deposit(self, amount):
12.         self.balance = self.balance + amount
13.
14.     def commit(self):
15.         with open(self.filepath, 'w') as file:
16.             file.write(str(self.balance))
17.
18. account = Account("account/balance.txt")
19. print(account.balance)
20. account.deposit(100)
21. print(account.balance)
22. account.commit()
```

balance.py

```
1000
```

* Inheritance.

```
class Account:
    def __init__(self, filepath):
        self.filepath = filepath
        with open(filepath, 'r') as file:
            self.balance = int(file.read())
```

balance.py

```
1000
```

```
def withdraw(self, amount):
    self.balance = self.balance - amount

def deposit(self, amount):
    self.balance = self.balance + amount

def commit(self):
    with open(self.filepath, 'w') as fobj:
        fobj.write(str(self.balance))

class Checking(Account):          # inheritance
    def __init__(self, filepath, fee):
        Account.__init__(self, filepath)
        self.fee = fee
    def transfer(self, amount):
        self.balance = self.balance - amount - self.fee

checking = Checking("account\\balance.txt", 1)
checking.transfer(100)
print(checking.balance)
checking.commit()
```

* OOP glossary
  - class
  - object instance
  - instance variable
  - class variable
  - DOC strings
  - Data member.
  - Constructor :- constructs class.
  - Methods
  - Instantiation.
  - Inheritance :- subclass out of base class
  - Attributes.