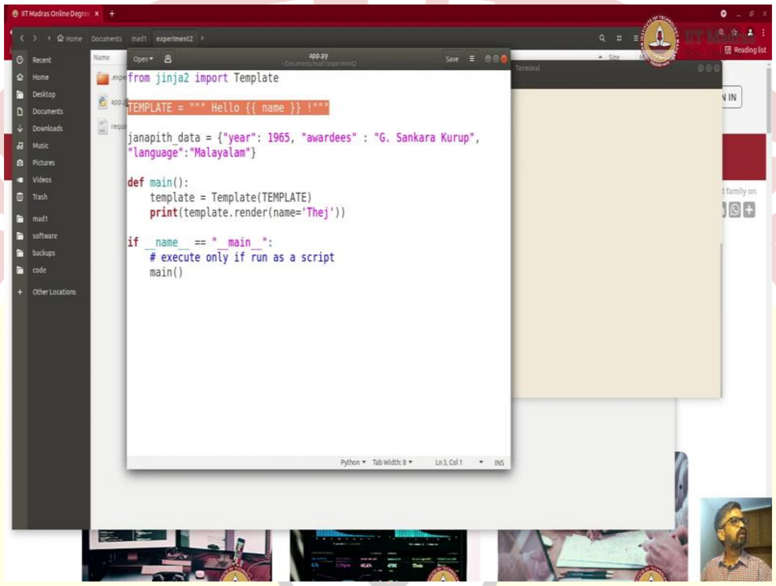
INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

IIT Madras

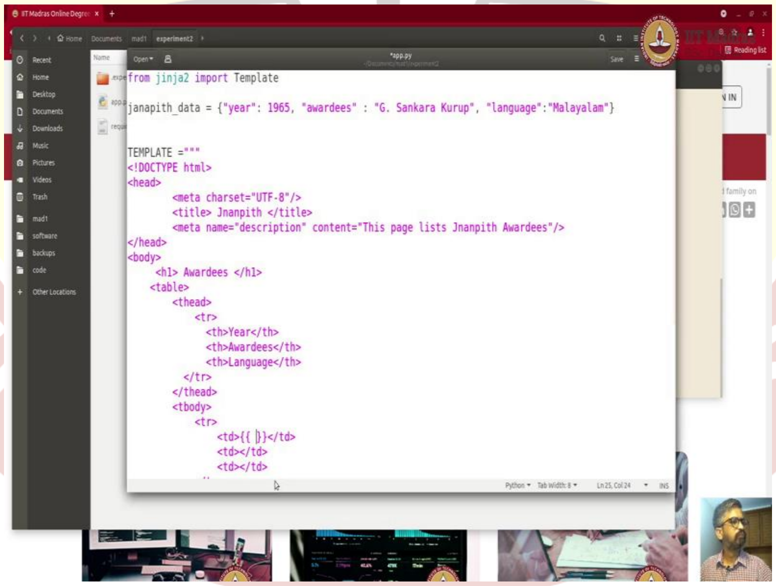
ONLINE DEGREE

Modern Application Development – I
Professor. Thejesh G N
Software Consultant
Indian Institute of Technology, Madras
Introduction to Jinja2 - II

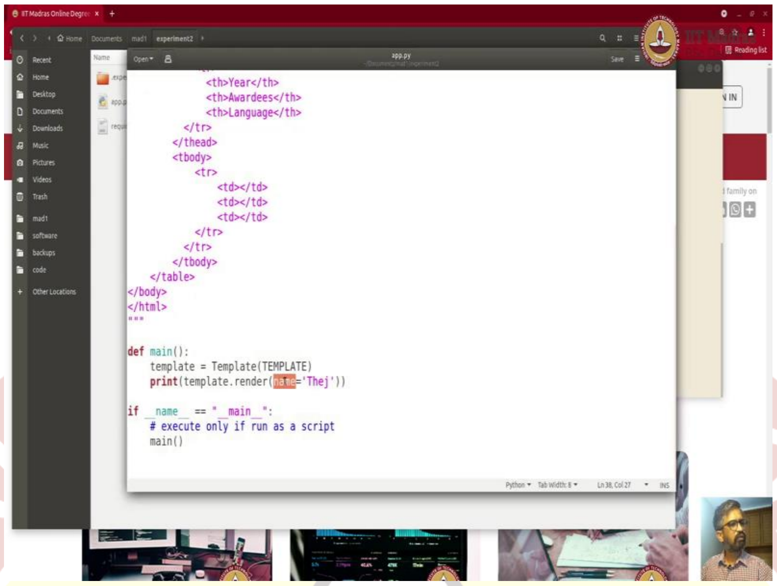
(Refer Slide Time: 0:16)



```
from jinja2 import Template

TEMPLATE = """ Hello {{ name }} !"""

janapith_data = {"year": 1965, "awardees" : "G. Sankara Kurup",
"language":"Malayalam"}

def main():
    template = Template(TEMPLATE)
    print(template.render(name='Thej'))

if __name__ == "__main__":
    # execute only if run as a script
    main()
```



```
from jinja2 import Template

janapith_data = {"year": 1965, "awardees" : "G. Sankara Kurup", "language":"Malayalam"}


TEMPLATE ="""
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ }}</td>
                <td></td>
                <td></td>
```


```
                <th>Year</th>
                <th>Awardees</th>
                <th>Language</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td></td>
                <td></td>
                <td></td>
            </tr>
        </tr>
        </tbody>
    </table>
</body>
</html>
"""

def main():
    template = Template(TEMPLATE)
    print(template.render(name='Thej'))

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

Now let us take a problem, let us continue from where we left where we had a table of janapith bodies. Now let us assume that we had a data set of janapith bodies and we wanted to create an html document out of it. Let us say the data is in a dictionary format, say here is the data right the data has only one row currently. It has year, awardee's name and which language yes this is the thing.

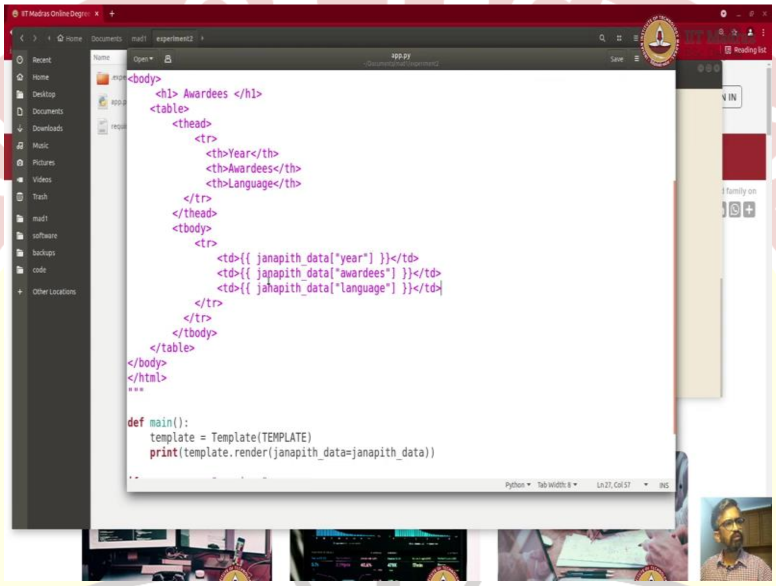
I want to create a html document similar to what we had created in experiment 1, right; it has a table and a table will show the columns. Now let us do that by creating a html document in a template. Now what you could do is uh let me just show you the like you know how the final template would look like and then we can do a substitution right. Let me just create a simple html actually I am just going to paste it okay I am going to replace this template with the template of html.

I am going to show you the value here let me just make it a little bigger okay, you can see here the template has all the html documentation; head part you know basic tags required then it has a awardees header then it has a table and a row but the row does not have any values. Now we want to populate these row values from this actual values into this so that when I do our template dot render template I get that but to get that I want to pass this you know data right to the template.

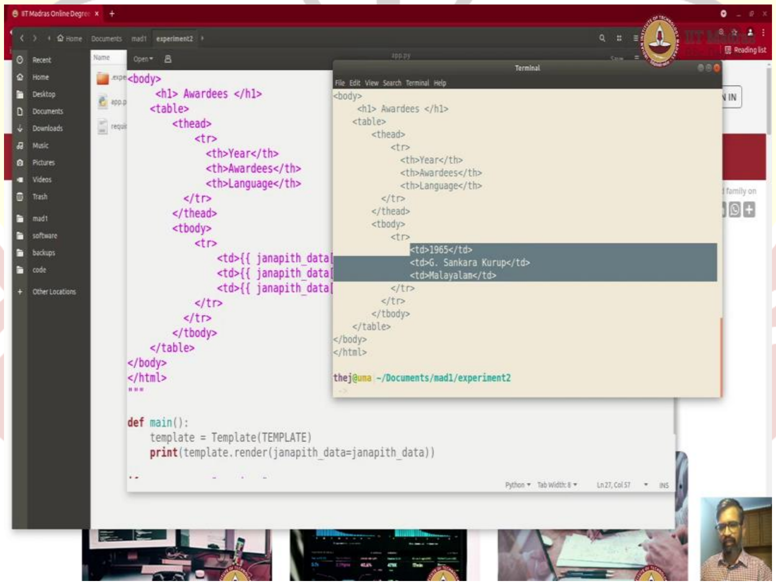
Let us try and pass it, let us try and pass it. Now if i run this it just prints, it prints the same thing there is no substitution happening here. So, for the substitution happen we have to use this janapith data and insert those subs like variables here. Let us do that you know like we saw you know double quote and here the data is a dictionary and the values are inside the dictionary.

So you can access it like you would access it in python you know have it a square bracket and then year so that it prints in here.

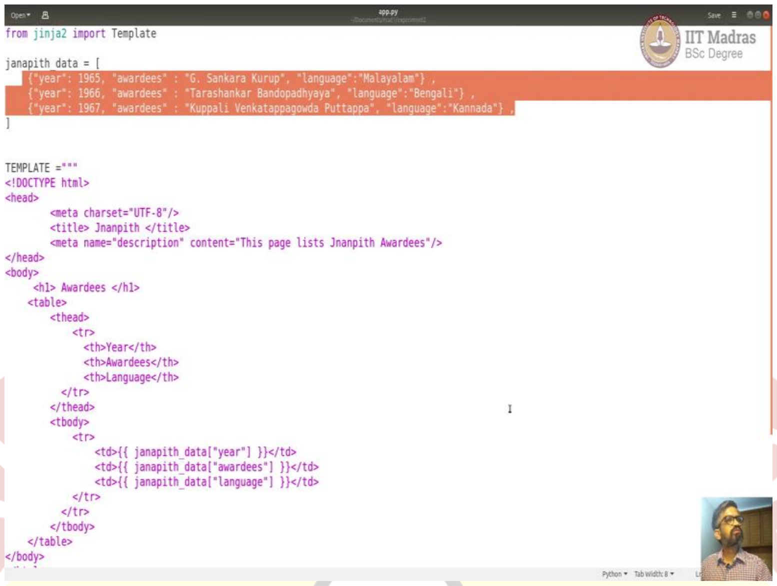
(Refer Slide Time: 3:21)



```
<body>
    <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
                <th>Year</th>
                <th>Awardees</th>
                <th>Language</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ janapith_data["year"] }}</td>
                <td>{{ janapith_data["awardees"] }}</td>
                <td>{{ janapith_data["language"] }}</td>
            </tr>
          </tr>
        </tbody>
    </table>
</body>
</html>
"""

def main():
    template = Template(TEMPLATE)
    print(template.render(janapith_data=janapith_data))
```



```
<body>
    <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
                <th>Year</th>
                <th>Awardees</th>
                <th>Language</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1965</td>
                <td>G. Sankara Kurup</td>
                <td>Malayalam</td>
            </tr>
          </tr>
        </tbody>
    </table>
</body>
</html>

thej@uma:~/Documents/mad1/experiment2
```



```
from jinja2 import Template

janapith_data = [
    {"year": 1965, "awardees" : "G. Sankara Kurup", "language":"Malayalam"} ,
    {"year": 1966, "awardees" : "Tarashankar Bandopadhyaya", "language":"Bengali"} ,
    {"year": 1967, "awardees" : "Kuppali Venkatappagowda Puttappa", "language":"Kannada"} ,
]


TEMPLATE ="""
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ janapith_data["year"] }}</td>
                <td>{{ janapith_data["awardees"] }}</td>
                <td>{{ janapith_data["language"] }}</td>
            </tr>
          </tr>
        </tbody>
    </table>
</body>
```

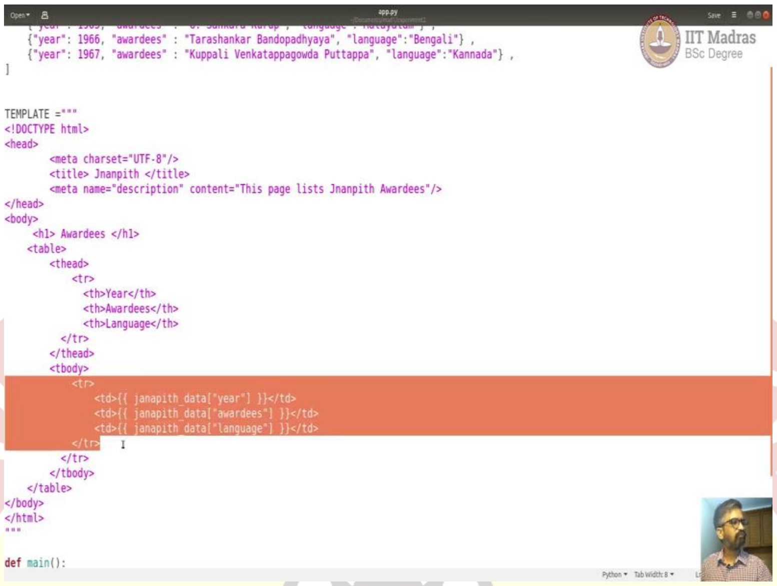
Similarly we can do for you know awardee's and so I am just going to copy paste and modify it right, I am just going to replace for awardees and language right. If I save now does it look good yes it does look good okay let us run now. If I run now; now you can see that in our html document we are actually replacing the values now this is okay if there was only one you know only one row.

What if we had many rows? Assume that we had a collection of or list of rows or list of dictionaries. Let us take one example here. Now here in this we have 3 rows right if you want this ideally should display 3 rows let us see what happens I am just passing janapith data here as this let us see what happens if I run now right.

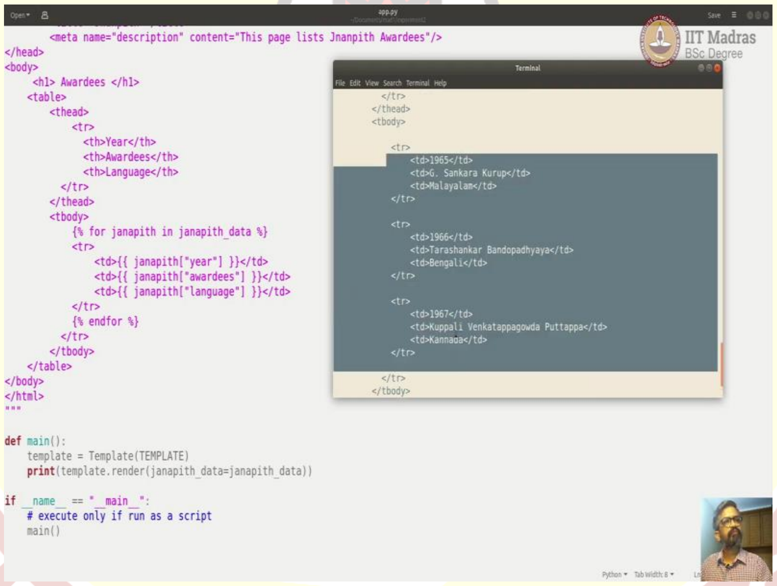
It does not do anything right because it is trying to assume this is a dictionary and trying to get values from it but this is actually not a dictionary this is a array. So, there is no substitution happening so now actually what we need to do is actually in a regular world will it trade through this array and then take one by one and create rows right.

So, we would probably put a for loop and then you know go around doing that creating rows for every loop.

(Refer Slide Time: 5:22)



```
    {"year": 1966, "awardees" : "Tarashankar Bandopadhyaya", "language":"Bengali"} ,
    {"year": 1967, "awardees" : "Kuppali Venkatappagowda Puttappa", "language":"Kannada"} ,
]


TEMPLATE ="""
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ janapith_data["year"] }}</td>
                <td>{{ janapith_data["awardees"] }}</td>
                <td>{{ janapith_data["language"] }}</td>
            </tr>
          </tr>
        </tbody>
    </table>
</body>
</html>
"""

def main():
```



```
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            {% for janapith in janapith_data %}
            <tr>
                <td>{{ janapith["year"] }}</td>
                <td>{{ janapith["awardees"] }}</td>
                <td>{{ janapith["language"] }}</td>
            </tr>
            {% endfor %}
          </tr>
        </tbody>
    </table>
</body>
</html>
"""

def main():
    template = Template(TEMPLATE)
    print(template.render(janapith_data=janapith_data))

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

```
          </tr>
        </thead>
        <tbody>

            <tr>
                <td>1965</td>
                <td>G. Sankara Kurup</td>
                <td>Malayalam</td>
            </tr>

            <tr>
                <td>1966</td>
                <td>Tarashankar Bandopadhyaya</td>
                <td>Bengali</td>
            </tr>

            <tr>
                <td>1967</td>
                <td>Kuppali Venkatappagowda Puttappa</td>
                <td>Kannada</td>
            </tr>

          </tr>
        </tbody>
```

So basically, if it was done manually you would have done roughly somewhat like this three rows of it right because there are three rows. But we would not do that like that we instead will use a for loop. Now Jinja also allows you to do a for loop similar to how you would do in python.

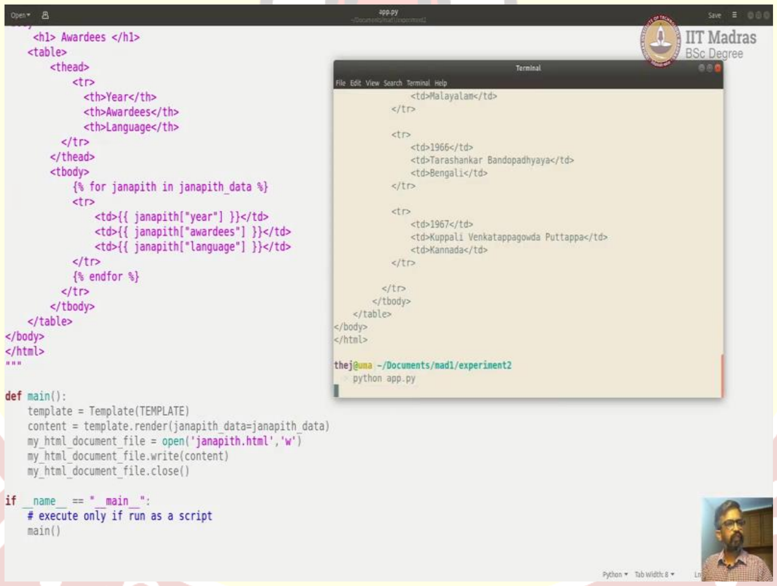
So, what I will do is I will write a for loop right, so now you can see that it looks different this has flower bracket and percentage this execute this part and same here but this double flower bracket actually gets replaced again here; here we are not trying to get from janapith because this is an array we are getting one janapith at a time from janapith data and trying to display.

So, this has to be these values okay. Now what we are doing is actually we are starting the follow we are taking one item janapith out of janapith data which is an array uh first one is let us say this and then we are trying to print here awardee's and language and we do that 3 times once that is done we end the for loop.

So, it actually if there are three rows it will create three rows here we are repeating the loop three times. Let us try and run this right, oh now you can see there is 3 rows here right created a html table dynamically with three rows with just this simple for loop. This is the power of Jinja2 you can do lot of complex things quite easily.

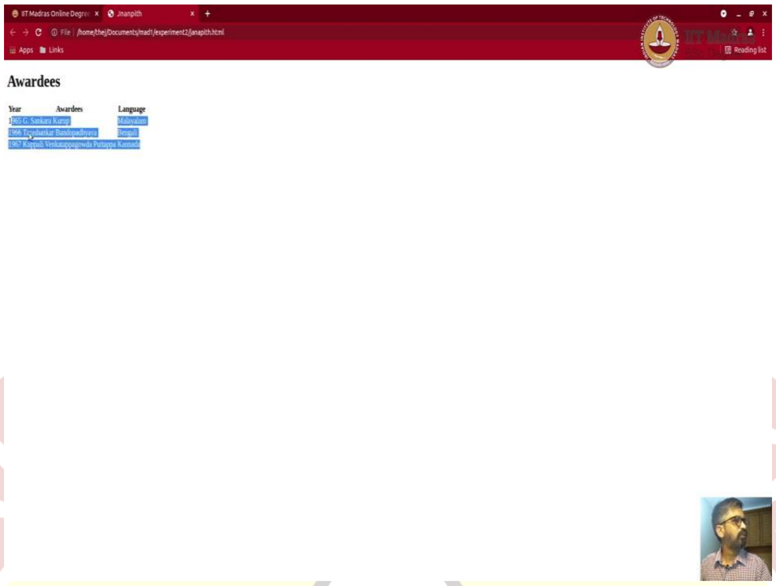
Now let us say after you render this you actually do want to print this right you want to actually write it to your file system or to a disk so that you can view the html, let us do that that is actually quite simple.

(Refer Slide Time: 7:46)



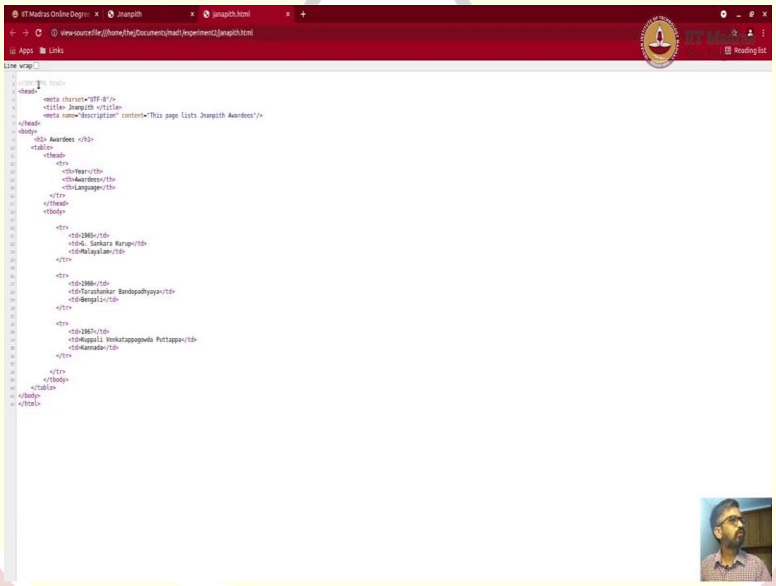
```
    <h1> Awardees </h1>
   <table>
       <thead>
           <tr>
             <th>Year</th>
             <th>Awardees</th>
             <th>Language</th>
          </tr>
       </thead>
       <tbody>
           {% for janapith in janapith_data %}
           <tr>
               <td>{{ janapith["year"] }}</td>
               <td>{{ janapith["awardees"] }}</td>
               <td>{{ janapith["language"] }}</td>
           </tr>
           {% endfor %}
          </tr>
       </tbody>
   </table>
</body>
</html>
"""

def main():
    template = Template(TEMPLATE)
    content = template.render(janapith_data=janapith_data)
    my_html_document_file = open('janapith.html','w')
    my_html_document_file.write(content)
    my_html_document_file.close()

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

```
                <td>Malayalam</td>
            </tr>

            <tr>
                <td>1966</td>
                <td>Tarashankar Bandopadhyaya</td>
                <td>Bengali</td>
            </tr>

            <tr>
                <td>1967</td>
                <td>Kuppali Venkatappagowda Puttappa</td>
                <td>Kannada</td>
            </tr>

          </tr>
        </tbody>
    </table>
</body>
</html>

thej@uma ~/Documents/mad1/experiment2
  python app.py
```



Awardees

Year	Awardees	Language
1965	G. Sankara Kurup	Malayalam
1966	Tarashankar Bandopadhyaya	Bengali
1967	Kuppali Venkatappagowda Puttappa	Kannada



```html
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>

            <tr>
                <td>1965</td>
                <td>G. Sankara Kurup</td>
                <td>Malayalam</td>
            </tr>

            <tr>
                <td>1966</td>
                <td>Tarashankar Bandopadhyaya</td>
                <td>Bengali</td>
            </tr>

            <tr>
                <td>1967</td>
                <td>Kuppali Venkatappagowda Puttappa</td>
                <td>Kannada</td>
            </tr>

          </tr>
        </tbody>
    </table>
</body>
</html>
```

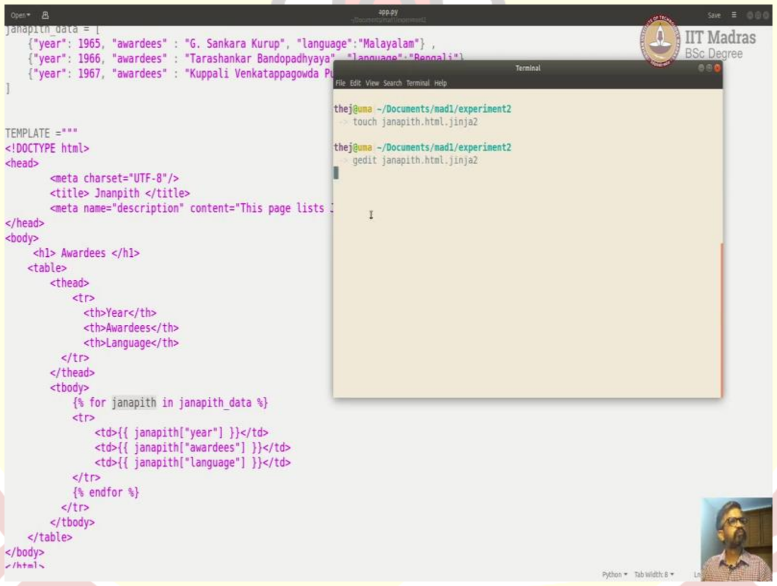
You can open up the file and then you can write the template. So I am going to show you I am just going to remove this, let us call this instead of printing it directly I will call it content. Then actually I will open a file called nanofit.html for writing purpose yeah So, I am opening a document file opening it file name is janapith.html for writing purpose. If the file does not exist it will create a new file.

Now I will to this opened file I will write the content and that is done just by writing content. Once I write the content I am going to close the file all right so now what we are doing? We have a list of janapith bodies we have a template which renders them as table once it is rendered I am going to open a local html file write that content and close it let us run this and see how it looks right. I did not run let us go to the thing so you can see it is been generated, the html file has been generated.

Let us go open that in the browser right here your table is created and you can see that you know all three rows exist if you want to view source you can view source it is clean html written like almost written using hand very nicely. Now usually you do not want to keep this template in a same code base.

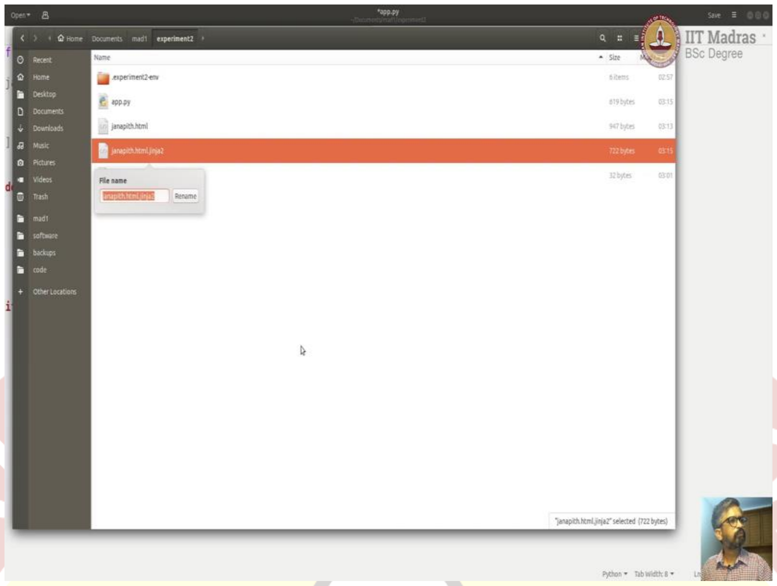
Now in the same file which you have code basically you would ideally try to keep it in a separate file to keep the content or the template separate from the code that generate uses the template yes; so I instead of actually keeping it here I will actually create a separate file called template file and I will write it to it. You can actually name with any extension.

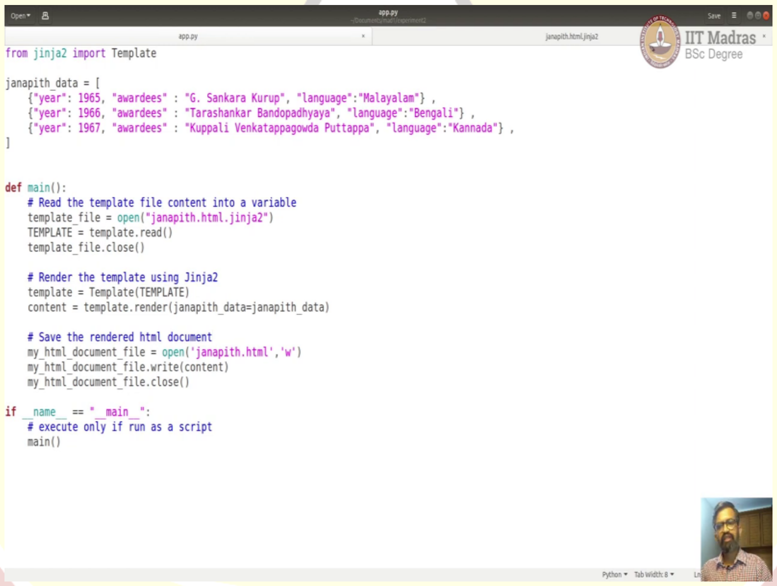
(Refer Slide Time: 10:48)



```
janapith_data = [
    {"year": 1965, "awardees" : "G. Sankara Kurup", "language":"Malayalam"} ,
    {"year": 1966, "awardees" : "Tarashankar Bandopadhyaya", "language":"Bengali"},
    {"year": 1967, "awardees" : "Kuppali Venkatappagowda Pu
]


TEMPLATE ="""
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists J
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            {% for janapith in janapith_data %}
            <tr>
                <td>{{ janapith["year"] }}</td>
                <td>{{ janapith["awardees"] }}</td>
                <td>{{ janapith["language"] }}</td>
            </tr>
            {% endfor %}
          </tr>
        </tbody>
    </table>
</body>
</html>
```

```
thej@uma ~/Documents/mad1/experiment2
 -> touch janapith.html.jinja2

thej@uma ~/Documents/mad1/experiment2
 -> gedit janapith.html.jinja2
```





```python
from jinja2 import Template

janapith_data = [
    {"year": 1965, "awardees" : "G. Sankara Kurup", "language":"Malayalam"} ,
    {"year": 1966, "awardees" : "Tarashankar Bandopadhyaya", "language":"Bengali"} ,
    {"year": 1967, "awardees" : "Kuppali Venkatappagowda Puttappa", "language":"Kannada"} ,
]


def main():
    # Read the template file content into a variable
    template_file = open("janapith.html.jinja2")
    TEMPLATE = template.read()
    template_file.close()

    # Render the template using Jinja2
    template = Template(TEMPLATE)
    content = template.render(janapith_data=janapith_data)

    # Save the rendered html document
    my_html_document_file = open('janapith.html','w')
    my_html_document_file.write(content)
    my_html_document_file.close()

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

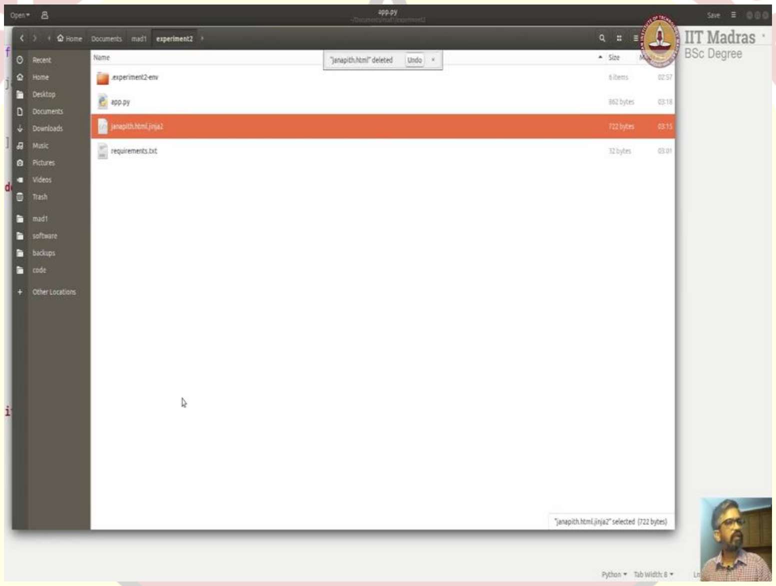
But I am going to create a file with one specific extension called you know template.html.Jinja2 or you know we can even call it because it is for janapith file we can call it janapith.html.jinja2. Then I will open it so here we have opened that file. What I will do now I will transfer this content into that independent template file so that code and template is separate I moved into a template file. I do not need this variable anymore but I need to read the template content.

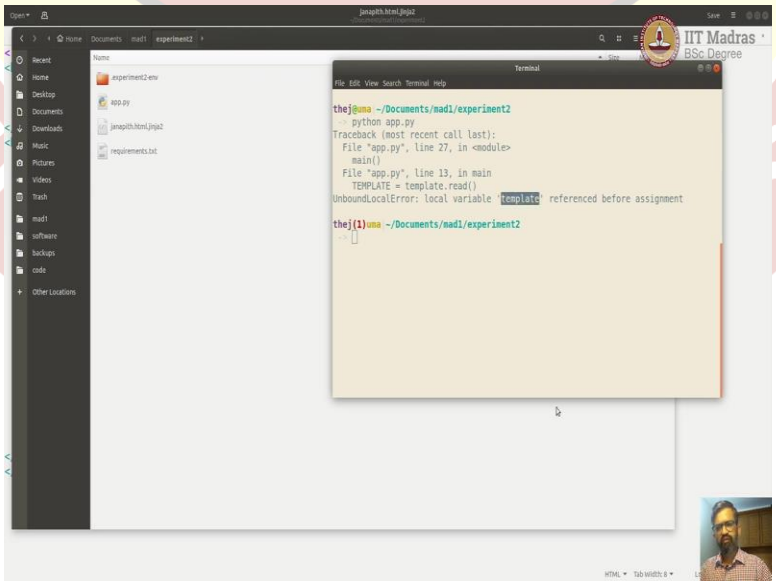
So basically, now I am going to read that file from the system and use that here okay let us open a template file. It is just generally used done using the same open command. Now open our template file name is janapith.html.Jinja2 just going copy it and paste it here I am just going to open this for read only purpose so you can just leave it or you can mark it read only.

Once the file is open I want to read the content of the file into a variable. Let us see same thing let us use the same thing template; template.read right once it is read I want to close this template file right close. Then that template content is passed on to our template class and then template object is created. So, here are the three part; first part I am just going to write some comment comments usually begin with hash.

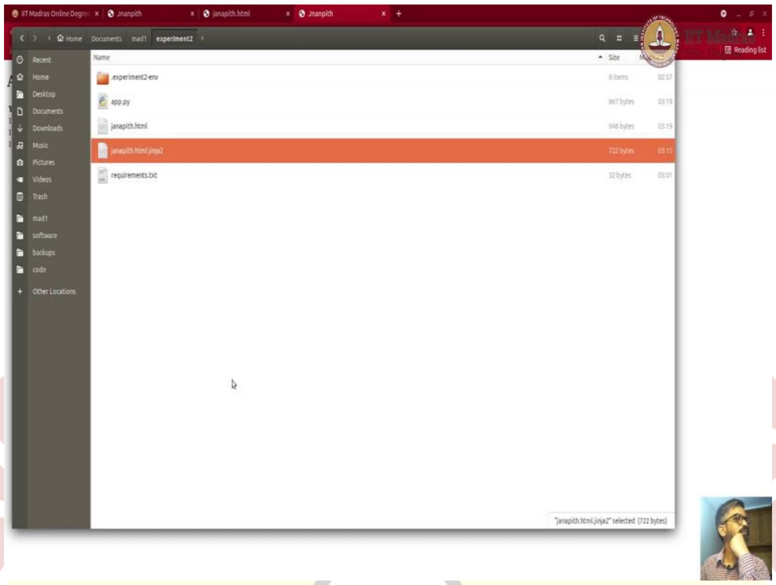
Read the template file content into a variable render the template using Jinja2, then save the rendered html document okay these are the three major steps that we are doing in this.

(Refer Slide Time: 14:20)





```
thej@uma ~/Documents/mad1/experiment2
 -> python app.py
Traceback (most recent call last):
  File "app.py", line 27, in <module>
    main()
  File "app.py", line 13, in main
    TEMPLATE = template.read()
UnboundLocalError: local variable 'template' referenced before assignment

thej(1)uma ~/Documents/mad1/experiment2
 ->
```



Name	Size	
.experiment2-env	6 items	02:57
app.py	867 bytes	03:19
janapith.html	946 bytes	03:19
janapith.html.jinja2	722 bytes	03:15
requirements.txt	32 bytes	03:01

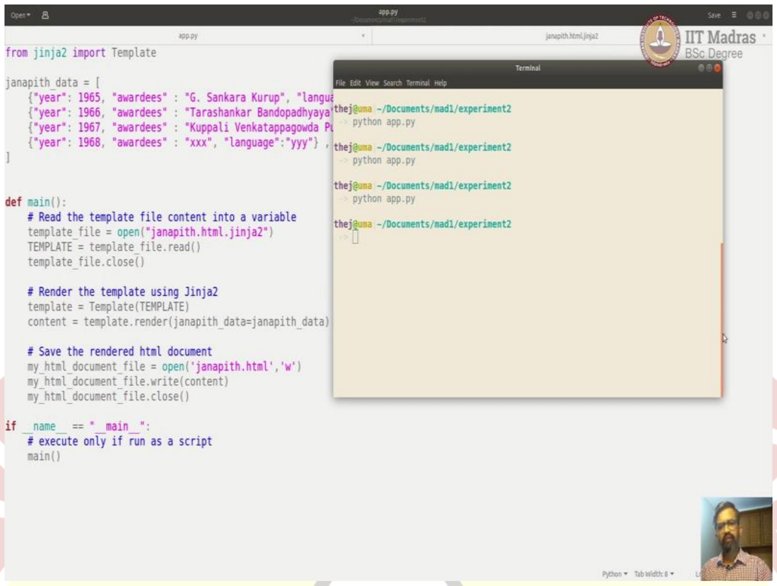
"janapith.html.jinja2" selected (722 bytes)

So, now let us just delete the previously generated janapith.html and re-run it, I opened it but rather clear it python and not by I am just going to open this folder so you can see as it generates, yes there is an error, okay.

Let us see what is the error, template file template file dot close it is not template.read it is actually template_file.read it is just a bug usually here you are opening the file, you are reading from the same file right so it has to be same, clear all right it is running. Will do it again I am just going to delete this, right here it is generated you can open it; generate the same file.

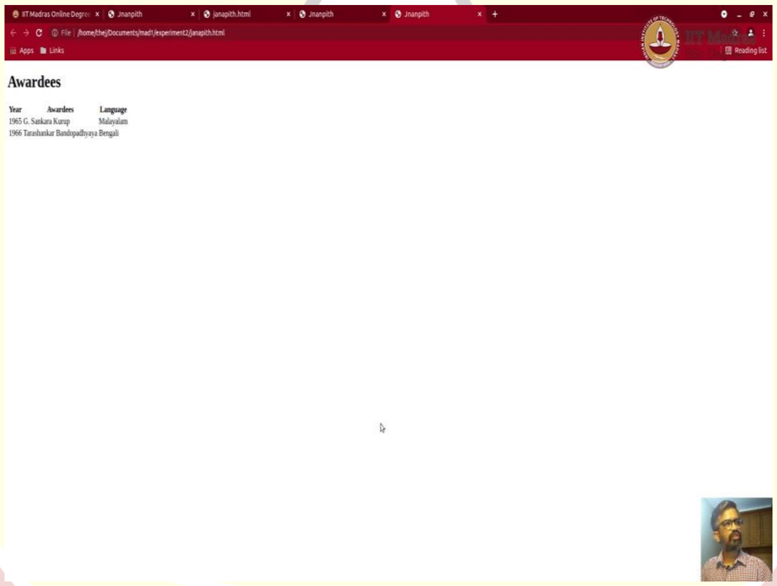
So, this way now if you see our thing we have a simple but very good a project structure where our application is in app.py our template is in template file requirements all the package requirements are in requirements.txt and we have all local virtually this is the simplest project structure that you would find and we are generating with our html dynamically.

(Refer Slide Time: 16:10)



```
from jinja2 import Template

janapith_data = [
    {"year": 1965, "awardees" : "G. Sankara Kurup", "langua
    {"year": 1966, "awardees" : "Tarashankar Bandopadhyaya'
    {"year": 1967, "awardees" : "Kuppali Venkatappagowda Pu
    {"year": 1968, "awardees" : "xxx", "language":"yyy"} ,
]


def main():
    # Read the template file content into a variable
    template_file = open("janapith.html.jinja2")
    TEMPLATE = template_file.read()
    template_file.close()

    # Render the template using Jinja2
    template = Template(TEMPLATE)
    content = template.render(janapith_data=janapith_data)

    # Save the rendered html document
    my_html_document_file = open('janapith.html','w')
    my_html_document_file.write(content)
    my_html_document_file.close()

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

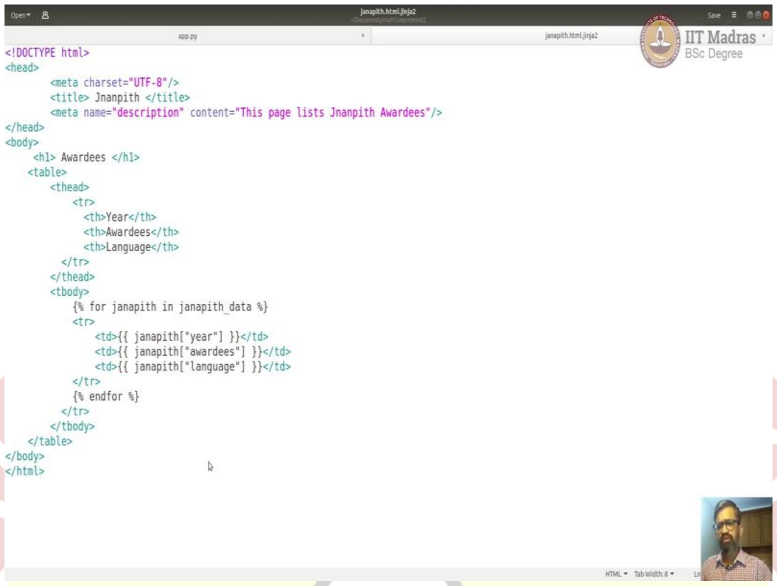
```
thej@uma ~/Documents/mad1/experiment2
 -> python app.py

thej@uma ~/Documents/mad1/experiment2
 -> python app.py

thej@uma ~/Documents/mad1/experiment2
 -> python app.py

thej@uma ~/Documents/mad1/experiment2
 ->
```



/home/thej/Documents/mad1/experiment2/janapith.html

Awardees

```
Year      Awardees        Language
1965 G. Sankara Kurup      Malayalam
1966 Tarashankar Bandopadhyaya Bengali
```

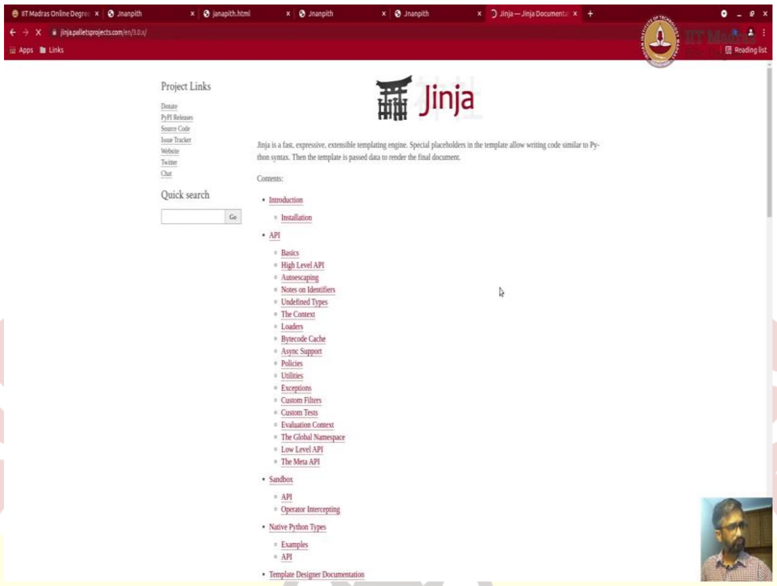


```
<!DOCTYPE html>
<head>
        <meta charset="UTF-8"/>
        <title> Jnanpith </title>
        <meta name="description" content="This page lists Jnanpith Awardees"/>
</head>
<body>
     <h1> Awardees </h1>
    <table>
        <thead>
            <tr>
              <th>Year</th>
              <th>Awardees</th>
              <th>Language</th>
          </tr>
        </thead>
        <tbody>
            {% for janapith in janapith_data %}
            <tr>
                <td>{{ janapith["year"] }}</td>
                <td>{{ janapith["awardees"] }}</td>
                <td>{{ janapith["language"] }}</td>
            </tr>
            {% endfor %}
          </tr>
        </tbody>
    </table>
</body>
</html>
```

Now you know if we add one more row let us say 1968 I do not know who got it in 1968 let us say xxx uh yeah and language yyy I do not know who with who got in 1968 so I am just going with random. So, I am going to re-run and open this again you can see the new got added so you know for loop is working quite well, it is the same thing way if you remove you know two of them and then you re-run and you can refresh this you can see it is changed right.

Now we can do many other things you can add a style to your template file, just the way you would have added in html and that would work too. This is the just like a simple template using Jinja2 you could assume this is just a taste of Jinja2, Jinja2 is very powerful you can do many other things with Jinja2 including like in python how would you do conditional statements, if else similar conditional statements you could do, you could do macros you could do blocks, you could do template hierarchy, have a base template, then extend that base template or you can also include one template into an another template so that you could you reuse the code.

(Refer Slide Time: 18:09)



Project Links

Donate
PyPI Releases
Source Code
Issue Tracker
Website
Twitter
Chat

Quick search

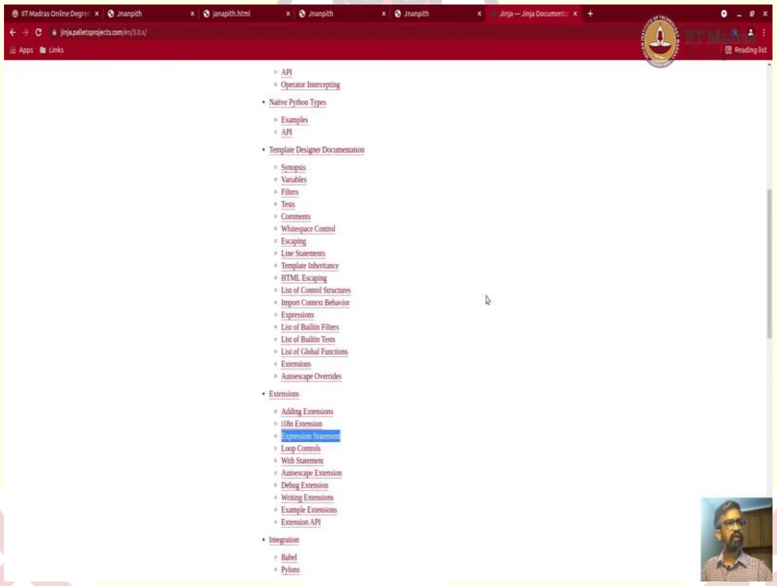
Go

Jinja

Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

Contents:

- Introduction
  - Installation
- API
  - Basics
  - High Level API
  - Autoescaping
  - Notes on Identifiers
  - Undefined Types
  - The Context
  - Loaders
  - Bytecode Cache
  - Async Support
  - Policies
  - Utilities
  - Exceptions
  - Custom Filters
  - Custom Tests
  - Evaluation Context
  - The Global Namespace
  - Low Level API
  - The Meta API
- Sandbox
  - API
  - Operator Intercepting
- Native Python Types
  - Examples
  - API
- Template Designer Documentation



  - API
  - Operator Intercepting
- Native Python Types
  - Examples
  - API
- Template Designer Documentation
  - Synopsis
  - Variables
  - Filters
  - Tests
  - Comments
  - Whitespace Control
  - Escaping
  - Line Statements
  - Template Inheritance
  - HTML Escaping
  - List of Control Structures
  - Import Context Behavior
  - Expressions
  - List of Builtin Filters
  - List of Builtin Tests
  - List of Global Functions
  - Extensions
  - Autoescape Overrides
- Extensions
  - Adding Extensions
  - i18n Extension
  - Expression Statement
  - Loop Controls
  - With Statement
  - Autoescape Extension
  - Debug Extension
  - Writing Extensions
  - Example Extensions
  - Extension API
- Integration
  - Babel
  - Pylons

And I will show you the page or where you can get more information about Jinja2; go to Jinja2 project from pelletsproject.com uh which is the project home or you could just search it on your search, service Google or Bing for Jinja2 and you should land up here and then you could explore all the things that you can do with it for example like the way i was saying template inheritance.

You could do like you know html escaping or you could do like loop controls, expressions many other things I think you should uh explore it when you get time it is quite interesting and you could generate all kinds of documents using this. That is it for today. Thank you so much. Bye.