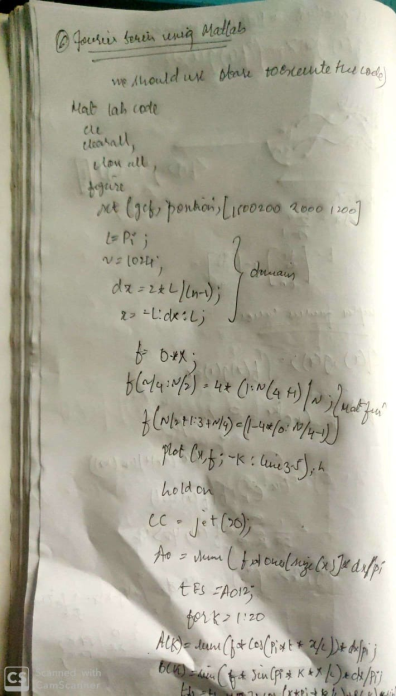
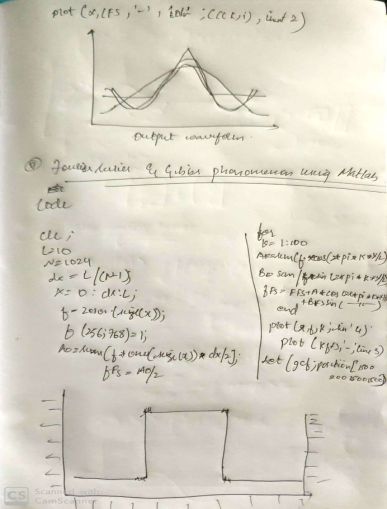
**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **25-May-2020** | **Name:** | **Russell D’souza** |
| **Course:** | **DSP** | **USN:** | **4AL15EC023** |
| **Topic:** | **Fouriers series** | **Semester & Section:** | **8th sem & ‘A’ section** |
| **Github Repository:** | **Russell1005** |  |  |

|  |
| --- |
| **MORNING SESSION DETAILS** |
| **Image of session**  **E:\course\ankitha-course\picture\IMG-20200523-WA0036.jpg E:\course\ankitha-course\picture\IMG-20200523-WA0018.jpg**  **E:\course\ankitha-course\picture\IMG-20200523-WA0035.jpg E:\course\ankitha-course\picture\IMG-20200523-WA0034.jpg E:\course\ankitha-course\picture\IMG-20200523-WA0033.jpg**  **E:\course\ankitha-course\picture\IMG-20200523-WA0032.jpg** E:\course\ankitha-course\picture\IMG-20200523-WA0031.jpgE:\course\ankitha-course\picture\IMG-20200523-WA0021.jpg  **Fourier series**     1. Fourier Series – Part 1 and part-2.     2)Inner Product in Hilbert Transform & Complex Fourier Series |

3) Fourier Series using Matlab

  4)Gibbs Phenomena UsingMatlab

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Date:** | **25-5-2020** | **Name:** | **Russell D’souza** | | **Course:** | **Python programming** | **USN:** | **4AL15EC023** | | **Topic:** | **Programming errors, build a website blocker** | **Semester & Section:** | **8th A** | | **Github Repository:** | **Russell1005** |  |  | |
|  |
|  |
| **AFTERNOON SESSION DETAILS** | |
| **Image of session** | |

## Errors

Errors or mistakes in a program are often referred to as bugs. They are almost always the fault of the programmer. The process of finding and eliminating errors is called debugging. Errors can be classified into three major groups:

* Syntax errors
* Runtime errors
* Logical errors

### Syntax errors

Python will find these kinds of errors when it tries to parse your program, and exit with an error message without running anything. Syntax errors are mistakes in the use of the Python language, and are analogous to spelling or grammar mistakes in a language like English: for example, the sentence Would you some tea? does not make sense – it is missing a verb.

Common Python syntax errors include:

* leaving out a keyword
* putting a keyword in the wrong place
* leaving out a symbol, such as a colon, comma or brackets
* misspelling a keyword
* incorrect indentation
* empty block

**Runtime errors**

If a program is syntactically correct – that is, free of syntax errors – it will be run by the Python interpreter. However, the program may exit unexpectedly during execution if it encounters a *runtime error* – a problem which was not detected when the program was parsed, but is only revealed when a particular line is executed. When a program comes to a halt because of a runtime error, we say that it has crashed.

Consider the English instruction *flap your arms and fly to Australia.* While the instruction is structurally correct and you can understand its meaning perfectly, it is impossible for you to follow it.

Some examples of Python runtime errors:

* division by zero
* performing an operation on incompatible types
* using an identifier which has not been defined
* accessing a list element, dictionary value or object attribute which doesn’t exist
* trying to access a file which doesn’t exist

Runtime errors often creep in if you don’t consider all possible values that a variable could contain, especially when you are processing user input. You should always try to add checks to your code to make sure that it can deal with bad input and edge cases gracefully. We will look at this in more detail in the chapter about exception handling.

**Logical errors**

Logical errors are the most difficult to fix. They occur when the program runs without crashing, but produces an incorrect result. The error is caused by a mistake in the program’s logic. You won’t get an error message, because no syntax or runtime error has occurred. You will have to find the problem on your own by reviewing all the relevant parts of your code – although some tools can flag suspicious code which looks like it could cause unexpected behaviour.

Sometimes there can be absolutely nothing wrong with your Python implementation of an algorithm – the algorithm itself can be incorrect. However, more frequently these kinds of errors are caused by programmer carelessness. Here are some examples of mistakes which lead to logical errors:

* using the wrong variable name
* indenting a block to the wrong level
* using integer division instead of floating-point division
* getting operator precedence wrong
* making a mistake in a boolean expression
* off-by-one, and other numerical errors

If you misspell an identifier name, you may get a runtime error or a logical error, depending on whether the misspelled name is defined.

A common source of variable name mix-ups and incorrect indentation is frequent copying and pasting of large blocks of code. If you have many duplicate lines with minor differences, it’s very easy to miss a necessary change when you are editing your pasted lines. You should always try to factor out excessive duplication using functions and loops – we will look at this in more detail later.

**Scheduling a Python program on a 24/7 server**

Keeping your computer on 24-7 is not practical, so if you want to execute a Python script at a particular time every day, you probably need a computer that is on all the time.

PythonAnywhere gives you access to such a 24-7 computer. You can upload a Python script and schedule it to run at a certain time every day. This availability can be useful, for example, when you want to extract some values (e.g., weather data) from a website and generate a text file with the value or other reports every day.

To schedule a Python script for execution on PythonAnywhere, follow these simple steps:

1. Sign up for a free account at https://www.pythonanywhere.com.
2. Go to your Dashboard, Files, Upload a File, and upload the Python file you want to schedule for execution.
3. Go to Tasks and set the time of the day you want your script to be executed and type in the name of the Python file you  uploaded (e.g., myscript.py). Note that the time you enter should be in UTC.
4. Click the Create button and you’re done.

Your Python file will now be executed every day at your specified time. If you don't have a Python script and you’re still confused about the benefit of this, here is a very simple Python script that you can  use to try the above steps:

If you don’t have a Python script and you’re still confused about the benefits of this PythonAnywhere feature, here is a very simple Python script you can use to schedule for execution:

1. from datetime import datetime
2. with open(datetime.now().strftime("%Y-%m-%d-%H-%M-%S"), "w") as myfile:
3. myfile.write("Hi there!")

The above code creates a text file and writes the string “Hi there!”  in that text file. The name of the text file will be the current date and time. For example one file name example would be 2018-02-16-18-20-33.txt.

That name is generated by datetime.now() indicating the date and time the script was executed.  Every time the script is executed, the script generates a new text file with a different name. You will have a new text file created every day.

**The any() function**

Hi! In case you didn't understand the if not any(website in line for website in website\_list) part in the previous video, here is another example:

1. >>> lines  = ["trees are good", "pool is fresh", "face is round"]
2. >>> website\_list = ["face", "clock", "trend"]
3. >>> for line in lines:
4. ...     any(website in line for website in website\_list)
5. ...
6. False
7. False
8. True

We start iterating over the items of website\_list using a for loop. In the first iteration we would have:

any(website in "trees are good" for website in website\_list)

Inside the parenthesis of any() there's another loop that iterates over website\_list:

1. ("face" in "trees are good")
2. ("clock" in "trees are good")
3. ("trend" in "trees are good")

If any of the above is True you get the expression evaluated to True. In this case none of them is True, so you get False.

If you want to return True (if all of them are True), use all() instead of any().

So, the part any(website in line for website in website\_list) will either be equal to True or False.

**CODE FOR WEBSITE BLOCKER**

import time

from datetime import datetime as dt

hosts\_temp=r"D:\Dropbox\pp\block\_websites\Demo\hosts"

hosts\_path="/etc/hosts"

redirect="127.0.0.1"

website\_list=["www.facebook.com","facebook.com","dub119.mail.live.com","www.dub119.mail.live.com"]

while True:

if dt(dt.now().year,dt.now().month,dt.now().day,8)

< dt.now() < dt(dt.now().year,dt.now().month,dt.now().day,16):

print("Working hours...")

with open(hosts\_path,'r+') as file:

content=file.read()

for website in website\_list:

if website in content:

pass

else:

file.write(redirect+" "+ website+"\n")

else:

with open(hosts\_path,'r+') as file:

content=file.readlines()

file.seek(0)

for line in content:

if not any(website in line for website in website\_list):

file.write(line)

file.truncate()

print("Fun hours...")

time.sleep(5)