

LIVE WEATHER DESKTOP NOTIFIER

A PROJECT REPORT

In partial fulfillment of the requirements for the award of the
degree

B.Tech

Under the guidance of Mr. DIPON MONDAL

BY

NARAHARI ANJALI



VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

In association with



SDF Building, Module #132, Ground Floor, Salt Lake City,
GP Block, Sector V, Kolkata, West Bengal 700091

1. **Title of the Project:** - Live Weather Desktop Notifier
2. **Project Members:** - Narahari Anjali
3. **Name of the Guide:** - Mr. Dipon Mondal
4. **Address:** - Ardent CompuTech Pvt. Ltd (An ISO 9001:2015 Certified) SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091

Project Version Control History:

VERSION	FINAL PROJECT
AUTHOR	Narahari Anjali
DESCRIPTION	PROJECT REPORT
DATE OF SUBMISSION	16th JUNE, 2024

DECLARATION

I hereby declare that the project work presented in the project proposal entitled “Live Weather Desktop Notifier” partially fulfills the requirements for the award of the Bachelor of Science in Computer Science Engineering at Ardent Computech Pvt. Ltd, Salt Lake, Kolkata, West Bengal, is an authentic work carried out under the guidance of Mr. Dipon Mondal. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:16th June 2024

Name of the Student: Narahari Anjali

CERTIFICATE

This is to certify that this proposal for the minor project entitled “Live Weather Desktop Notifier” is a record of bonafide work carried out by Narahari Anjali under my guidance at ARDENT COMPUTECH PVT LTD. In my opinion, the report in its present form partially fulfills the requirements for the award of the degree of B.Tech and as per regulations of the ARDENT®. To the best of my knowledge, the results embodied in this report are original and worthy of incorporation in the present version.

Guide / Supervisor

MR. DIPON MONDAL

Project Engineer

Ardent CompuTech Pvt. Ltd (An ISO 9001:2015 Certified) SDF Building, Module #132,
Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

The success of any project depends largely on the encouragement and guidelines of many others. I am writing to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I want to show our honest appreciation to Mr. Dipon Mondal, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged by his valuable advice and constant inspiration; this project would not have materialized without his encouragement and guidance.

Words are inadequate in offering our thanks to the other trainees, project assistants, and other Ardent CompuTech Pvt. Ltd. members for their encouragement and cooperation in this project work. The guidance and support received from all the members was vital for the success of this project.

CONTENTS

- OVERVIEW
- HISTORY OF PYTHON
- FEATURES OF PYTHON
- PYTHON IDENTIFIERS
- PYTHON KEYWORDS
- INDENTATION
- PYTHON DATA TYPES
- DATA TYPE CONVERSION
- PYTHON FUNCTION
- PYTHON MODULES
- GRAPHICAL USER INTERFACE
- INTRODUCTION TO TKINTER
- TKINTER WIDGETS
- WIDGETS USED IN THE PROJECT
- ACTUAL CODES USED IN THE PROJECT
- OUTPUTS
- FUTURE SCOPE OF THE PROJECT
- CONCLUSION
- BIBLIOGRAPHY
- THANK YOU

OVERVIEW

Python is a high-level, general-purpose, and prevalent programming language. Python programming language (latest Python 3) is used in web development, Machine Learning applications, and all cutting-edge technology in the Software Industry. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. Python supports modules and packages, which encourages program modularity and code reuse. The Python Interpreter and the Extensive Standard Library are available in source or binary form without charge for all major platforms and can be freely distributed. Python programming language (The latest version of Python 3) is used in web development, machine learning applications, and all cutting-edge technology in the software industry.

HISTORY OF PYTHON

In the late 1980s, history was about to be written. It was at that time that I started working on Python. Soon after, Guido Van Rossum began its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) in the Netherlands. It was started as a hobby project because he was looking for an exciting project to keep him occupied during Christmas.

The programming language in which Python is said to have succeeded is ABC Programming Language, which interfaced with the Amoeba Operating System and had the exception handling feature. He had already helped create ABC earlier in his career and had seen some issues with ABC but liked most of the features. After that, what he did was very clever. He had taken the syntax of ABC and some of its good features. It came with many complaints, too, so he fixed those issues entirely and created a good scripting language that had removed all the flaws.

FEATURES OF PYTHON

1. Free and Open Source

Since it is open-source, the source code is also available to the public. So one can download it, use it, and share it.

2. Easy to code

Python is a high-level programming language. It is easier to learn the language than other languages like C, C#, Javascript, Java, etc. It is straightforward to code in Python, and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

Learning Python is quite simple. As was already established, Python's syntax is straightforward. The indentations define the code block rather than semicolons or brackets.

4. Object-Oriented Language

One of the critical features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in Python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture or manage the memory.

7. Large Community Support

Python has gained popularity over the years. The enormous StackOverflow community constantly answers our questions. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.

8. Easy to Debug

Excellent information for mistake tracing. One will be able to quickly identify and correct the majority of the program's issues once one understands how to interpret Python's error traces. By glancing at the code, one can determine what it is designed to perform.

9. Python is a Portable language.

Python language is also a portable language. For example, if we have Python code for Windows and want to run it on other platforms such as Linux, Unix, and Mac, we do not need to change it; we can run it on any platform.

10. Python is an Integrated language.

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. Like other languages, such as C, C++, Java, etc., compiling Python code is unnecessary, making debugging our code more accessible. The source code of Python is converted into an immediate form called bytecode.

12. Large Standard Library

Python has an extensive standard library that provides a rich set of modules and functions, so one does not have to write the code for everything—many libraries in Python, such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically typed language. That means a variable's type (int, double, long, etc.) is decided at run time, not in advance. Because of this feature, we don't need to specify the variable type.

14. Frontend and backend development

With a new project, py script, one can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help one do frontend development work in Python, such as javascript. Backend is the strong forte of Python. It's extensively used for this work cause of its frameworks like Django and Flask.

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. When given a value, the memory is automatically allocated to a variable at runtime. Developers do not need to write `int y = 18` if the integer value 15 is set to y. You may type `y=18`.

PYTHON IDENTIFIERS

An identifier is a user-defined name given to a variable, function, class, module, etc. The identifier is a combination of character digits and an underscore. They are case-sensitive, i.e., 'num' and 'Num' and 'NUM' are three different identifiers in Python. It is a good programming practice to give meaningful names to identifiers to make the code understandable.

Rules for Naming Python Identifiers:

- It cannot be a reserved Python keyword.
- It should not contain white space.
- It can be a combination of A-Z, a-z, 0-9, or underscore.
- It should start with an alphabet character or an underscore (_).
- It should not contain any unique character other than an underscore (_).

PYTHON KEYWORDS

Python Keywords are some predefined and reserved words in Python that have special meanings. Keywords are used to define the syntax of the coding. The keyword cannot be used as an identifier, function, or variable name. All the keywords in Python are written in lowercase except True and False. There are 35 keywords in Python 3.11.

Keywords	Description
and	This is a logical operator which returns true if both the operands are true else returns false.
or	This is also a logical operator which returns true if anyone operand is true else returns false.
not	This is again a logical operator it returns True if the operand is false else returns false.
if	This is used to make a conditional statement.
elif	Elif is a condition statement used with an if statement. The elif statement is executed if the previous conditions were not true.
else	Else is used with if and elif conditional statements. The else block is executed if the given condition is not true.
for	This is used to create a loop.
while	This keyword is used to create a while loop.
break	This is used to terminate the loop.

as	This is used to create an alternative.
def	It helps us to define functions.
lambda	It is used to define the anonymous function.
pass	This is a null statement which means it will do nothing.
return	It will return a value and exit the function.
True	This is a boolean value.
False	This is also a boolean value.
try	It makes a try-except statement.
with	The with keyword is used to simplify exception handling.
assert	This function is used for debugging purposes. Usually used to check the correctness of code
class	It helps us to define a class.

continue	It continues to the next iteration of a loop
del	It deletes a reference to an object.
except	Used with exceptions, what to do when an exception occurs
finally	Finally is used with exceptions, a block of code that will be executed no matter if there is an exception or not.
from	It is used to import specific parts of any module.
global	This declares a global variable.
import	This is used to import a module.
in	It's used to check whether a value is present in a list, range, tuple, etc.
is	This is used to check if the two variables are equal or not.
none	This is a special constant used to denote a null value or avoid. It's important to remember, 0, any empty container(e.g empty list) do not compute to None

nonlocal	It's declared a non-local variable.
raise	This raises an exception.
yield	It ends a function and returns a generator.
async	It is used to create asynchronous coroutine.
await	It releases the flow of control back to the event loop.

INDENTATION

Python indentation tells a Python interpreter that the statements belong to a particular code block. A block is a combination of all these statements. Block can be regarded as the grouping of statements for a specific purpose. Most programming languages like C, C++, and Java use braces { } to define a code block. Python uses indentation to highlight the blocks of code. Whitespace is used for indentation in Python. All statements with the same distance to the right belong to the same block of code. If a block has to be more deeply nested, it is simply indented further to the right. You can understand it better by looking at the following lines of code.

Example of Python Indentation:

- Statement (line 1), if condition (line 2), and statement (last line) belong to the same block, which means that after statement 1, if condition will be executed. If the condition becomes false, Python will jump to the last statement for execution.
- The nested if-else belongs to block 2, which means that if nested if becomes False, then Python will execute the statements inside the else condition.
- Statements inside nested if-else belong to block 3, and only one statement will be executed depending on the if-else condition.

PYTHON DATA TYPES

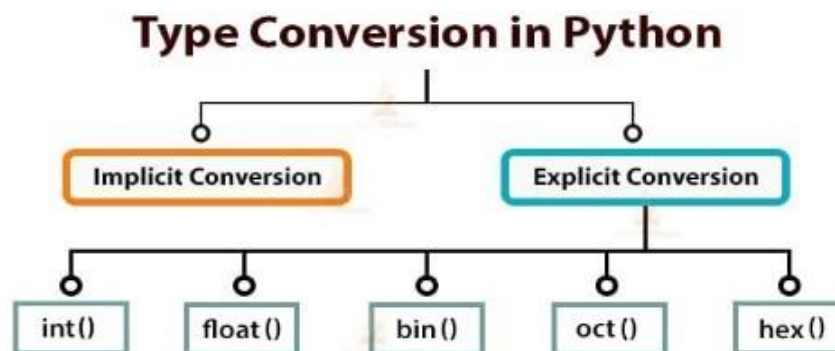
Python Data types are the classification or categorization of data items. It represents the value of what operations can be performed on a particular data. Since everything is an object in Python programming, Python data types are classes, and variables are instances (objects) of these classes. The following are the standard or built-in data types in Python:

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary
- Binary Types(memory view, byte array, bytes)

Data Types	Classes	Description
Numeric	int, float, complex	holds numeric values
String	str	holds sequence of characters
Sequence	list, tuple, range	holds collection of items
Mapping	dict	holds data in key-value pair form
Boolean	bool	holds either <code>True</code> or <code>False</code>
Set	set, frozenset	hold collection of unique items

DATA TYPE CONVERSION

Python defines type conversion functions to directly convert one data type to another, which is helpful in day-to-day and competitive programming. This article is aimed at providing information about certain conversion functions.



Sr.No.	Function & Description
1	int(x [,base]) Converts x to an integer. base specifies the base if x is a string
2	long(x [,base]) Converts x to a long integer. base specifies the base if x is a string.
3	float(x) Converts x to a floating-point number.
4	complex(real [,imag]) Creates a complex number.
5	str(x) Converts object x to a string representation.
6	repr(x) Converts object x to an expression string.
7	eval(str) Evaluates a string and returns an object.
8	tuple(s) Converts s to a tuple.
9	list(s) Converts s to a list.

PYTHON FUNCTION

Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code repeatedly for different inputs, we can do the function calls to reuse the code contained in it repeatedly.

Some Benefits of Using Functions:

- Increase Code Readability
- Increase Code Reusability

A function is a block of code that only runs when called. One can pass data, known as parameters, into a function. A function can return data as a result.

Example:

```
def my_function(fname, lname):  
  
    print(fname + " " + lname)  
  
my_function("Emil", "Refsnes")
```

1) PYTHON USER-DEFINED FUNCTIONS

All the functions written by any of us come under user-defined functions. Below are the steps for writing user-defined functions in Python.

- The “def” keyword in Python is used to declare user-defined functions.

- An indented block of statements follows the function name and arguments, which contain the function's body.

The constructional structure of a user-defined Python function is: - `def <function_name>([<parameters>]):`
`<statement(s)>` The components of the definition are explained In the table below:

SL . NO.	COMPONENETS	MEANING
1	Def	The keyword that informs Python that a function is being defined.
2	<function_name>	A valid Python identifier that names the function.
3	<parameters>	An optional, comma-separated list of parameters that may be passed to the function.
4	<statement(s)>	A block of valid Python statements. It is also called the body of the function. The body is a block of statements that will be executed when the function is called. The body of a Python function is defined by indentation in accordance with the off-side rule.
5	:	Punctuation that denotes the end of the Python function header (the name and parameter list).

There are five different aspects of user-defined functions as follows:

- 1) Parameterized Functions: - In a parameterized function, one or more details of what the function does are defined as parameters instead of being defined in the function; they have to be passed in by the calling code. The function may take arguments(s), also called parameters, as input within the opening and closing parentheses, just

after the function name followed by a colon. For example, `def myfunc(num1,num2,...numn): <statements to be executed>`

2) Functions with Default Arguments: - Python has a different way of representing syntax and default values for function arguments. Default values indicate that the function argument will take that value if no argument value is passed during the function call. The default value is assigned by using the assignment (=) operator. For example,

```
def myFun(x, y = 50):
```

```
    print("x = ", x)
```

```
    print("y = ", y)
```

```
myFun(10) # This is the driver code
```

This code will give the output like this: `x = 10 y = 50`. Here, only one parameter is passed, and it's assigned to `x`, and the value of `y` is assigned as its default value passed in the function declaration. Again, if we call the function like this:

```
def myFun(x, y = 50):
```

```
    print("x = ",x)
```

```
    print("y = ",y)
```

```
myFun(10,20) # This is also the driver code
```

This code will give the output like this:

`x = 10, y = 20`

Because values of both the parameters are passed, the default value is no longer required or valid.

- 3) Functions with Keyword Arguments: - When we call a function with some values, these values get assigned to the arguments according to their position. According to Wikipedia, in computer programming, named parameters, named arguments, or keyword arguments refer to a computer language's support for function calls that clearly state the name of each parameter within the function call. Python allows functions to be called using keyword arguments. When we call functions in this way, the order (position) of the arguments can be changed. For example,

```
def myFun(x,y): print("x = ", x) print("y = ", y)
```

```
myFun(10,20) # This is also the driver code
```

This code will give the output like this: `x = 10 y = 20`

```
def myFun(x,y): print("x = ", x) print("y = ", y)
```

```
myFun(y=20,x=10) # This is also the driver code
```

This code will also give the same output as the previous one: `x = 10, y = 20`

- 4) Pass by Reference and Pass by Value: One important thing to note is that in Python, every variable name is a reference. A new reference to the object is created when

we pass a variable to a function. Parameter passing in Python is the same as reference passing in Java.

2) Python Built-in Functions: The built-in functions are defined as those whose functionality is pre-defined in Python. The Python interpreter has several functions that are always present for use. These functions are known as Built-in Functions. The main advantage of built-in functions is that they are pre-defined in Python, so we can use them in our program or code wherever and whenever we want. As previously defined in Python, programmers can easily use them freely in their code, even without knowing the actual code or logic written inside the function; programmers must understand how and where to use them. There are many built-in functions defined in Python, which can be, by default, accessible by any programmer. Some of them can be listed as given in the below table: -

<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>_import_()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

3) Python Lambda Functions: Python allows us to create anonymous functions, i.e., functions having no names, using a facility called the Lambda function. Lambda functions are small functions that are usually not more than a line. It can have any number of arguments, just like a normal function. ... Also, there is no need for any return statement in lambda function. In Python, a lambda function is a single-line function declared with no name, which can have any number of arguments but can only have one expression. Such a function can behave similarly to a regular function declared using Python's def keyword. The usefulness of lambda will be realized when we need a small piece of function that will be run once in a while or just once. Instead of writing the function in the global scope or including it as part of our main program, we can toss around a few lines of code when needed for a variable or another function. It has syntax like lambda arguments: expression. For an example,

```
x = lambda a, b, c: a + b + c print(x(5, 6, 12))
```

This anonymous function evaluates the sum of all the numbers passed as arguments, i.e. $(5+6+12) = 23$. The output of this lambda expression will be: - 23

4) Python Recursive Functions: A recursive function is a function that calls itself during its execution. This enables the function to repeat itself several times, outputting the result at the end of each iteration. The process in which a function calls itself directly or indirectly is called recursion, and the corresponding function is called a recursive function. Using a

recursive algorithm, certain problems can be solved quite easily. This can be illustrated by the below example of finding the factorial of a number: -

```
def fact(n):  
    if n==1:  
        return 1  
    else:  
        return(n*fact(n-1))  
  
print('The factorial is: ',fact(int(input())) # This is the driver  
code
```

The output will be like this: suppose the user inputs 10:

The factorial is: 3628800

PYTHON MODULES

A module allows us to organize our Python code logically. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes you can bind and reference.

The Python code for a module named 'name' normally resides in a file named 'aname.py.' Here's an example of a simple module, support.py

```
def print_func( par ):  
  
    print("Hello: ", par)
```

The Import Statement

You can use any Python source file as a module by executing an import statement in another Python source file. The import has the following syntax – import module1[, module2[,... moduleN]

GRAPHIC USER INTERPHASE(GUI)

The standard and valuable component of electronics such as computers, tablets, and smartphones is a visual interface called a Graphical User Interface (GUI). The graphical user interface (GUI) shows valuable items that indicate actions the user can perform and transmit information as per requirement. When interacting with the objects, the user can modify or simplify their size, color, and visibility. As the text command-line interfaces in the system were complex and challenging to master, GUIs were developed. It is currently the accepted practice in software applications or system programming for required user-centered design.

COMPONENTS OF GUI:

- **Pointers:** The pointer appears on the user's screen as a marking symbol. The pointer moves on to choose instructions and objects as per requirement.
- **Icons:** Icons allude to tiny visual representations of windows, documents, actions, and other things on the display screen to simplify. A pointer and pointing device can be used by the user to carry out the initial tasks for the overall processes.
- **Pointing tool:** At the initial stages, the pointing tool enables the user to select and move the required pointer items on the screen, including a trackball or mouse. It is the most beneficial tool in GUI.
- **Desktop:** The desktop is the screen contained within the icons and user beneficial.

TKINTER

Python offers multiple options for developing GUI (Graphical User Interface). Tkinter is the most commonly used method of all the GUI methods. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python Tkinter is the fastest and easiest way to create GUI applications. Creating a GUI using Tkinter is an easy task.

To create a Tkinter Python app, you follow these basic steps:

- Import the tkinter module: This is done just like importing any other module in Python. Note that in Python 2. x, the module is named 'Tkinter,' while in Python 3. x, it is named 'tkinter'.
- Create the main window (container): The main window serves as the container for all the GUI elements you'll add later.
- Add widgets to the main window: You can add any number of widgets, like buttons, labels, entry fields, etc., to the main window to design the interface as desired.
- Apply event triggers to the widgets: You can attach them to the widgets to define how they respond to user interactions.

TKINTER WIDGETS

A Tkinter user interface is made up of individual widgets. Each widget is represented as a Python object, instantiated from classes; there are various pre-defined widgets available in Tkinter:

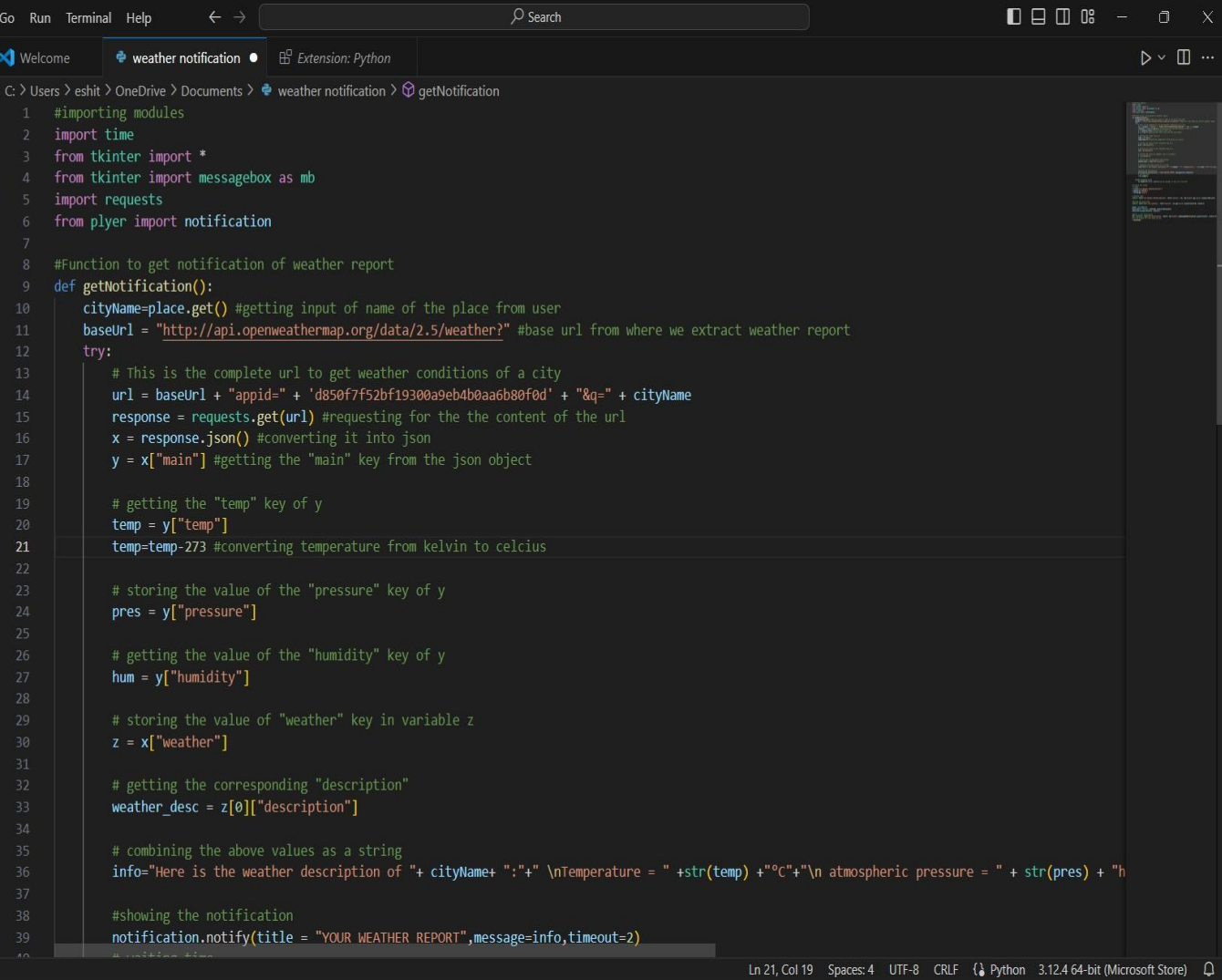
- Label: Used to contain text or images.
- Buttons: Similar to a Label but provides additional functionality for mouse hovers, presses, and releases, as well as keyboard activity/events.
- Canvas: Provides the ability to draw shapes (lines, ovals, polygons, rectangles); can contain images or bitmaps.
- Frame: Pure container for other widgets.
- Entry: Single-line text field with which to collect keyboard input.

and many more. We have used many such widgets in our project, including those mentioned above.

WIDGETS USED IN THE PROJECT

- `title()`: It displays the title on the top of the GUI.
- `config()`: It sets the background color of the window
- `geometry()`: It sets the length and width of the GUI.
- `Label()`: This helps in showing text on the window
- `Entry()`: This widget helps in taking input from the user
- `Button()`: This creates a button with the mentioned attributes, and the command parameter represents the function that executed on pressing the button
- `mainloop()`: This makes sure the screen runs till it is manually closed
- `get()`: It helps in getting the input given by the user in the `Entry()` widget
- `requests.get()`: Getting the data from the url
- `.json()`: converting data to .json format
- `notification.notify()`: showing the notification on desktop
- `showerror()`: Shows an error pop-up message when an exception occurs.

ACTUAL CODES USED IN THE PROJECT



The image shows a screenshot of a Visual Studio Code editor window. The top bar includes the 'Go' button, 'Run', 'Terminal', and 'Help' menus, along with a search bar. The active tab is 'weather notification', which is a Python extension. The file explorer on the left shows the project structure: 'C:\Users> eshit > OneDrive > Documents > weather notification > getNotification'. The main editor area displays a Python script for getting weather notifications. The script imports modules like 'time', 'tkinter', 'requests', and 'plyer'. It defines a 'getNotification()' function that takes a city name as input, constructs a URL to the OpenWeatherMap API, makes a GET request, and extracts the temperature, pressure, humidity, and weather description. Finally, it uses 'plyer.notification' to display a notification with the title 'YOUR WEATHER REPORT' and the extracted information. The status bar at the bottom indicates the current position is 'Ln 21, Col 19' with 'Spaces: 4', 'UTF-8' encoding, 'CRLF' line endings, and 'Python 3.12.4 64-bit (Microsoft Store)'.

```
1 #importing modules
2 import time
3 from tkinter import *
4 from tkinter import messagebox as mb
5 import requests
6 from plyer import notification
7
8 #Function to get notification of weather report
9 def getNotification():
10     cityName=place.get() #getting input of name of the place from user
11     baseUrl = "http://api.openweathermap.org/data/2.5/weather?" #base url from where we extract weather report
12     try:
13         # This is the complete url to get weather conditions of a city
14         url = baseUrl + "appid=" + 'd850f7f52bf19300a9eb4b0aa6b80f0d' + "&q=" + cityName
15         response = requests.get(url) #requesting for the the content of the url
16         x = response.json() #converting it into json
17         y = x["main"] #getting the "main" key from the json object
18
19         # getting the "temp" key of y
20         temp = y["temp"]
21         temp=temp-273 #converting temperature from kelvin to celcius
22
23         # storing the value of the "pressure" key of y
24         pres = y["pressure"]
25
26         # getting the value of the "humidity" key of y
27         hum = y["humidity"]
28
29         # storing the value of "weather" key in variable z
30         z = x["weather"]
31
32         # getting the corresponding "description"
33         weather_desc = z[0]["description"]
34
35         # combining the above values as a string
36         info="Here is the weather description of " + cityName+ " : "+" \nTemperature = " +str(temp) +"°C"+" \n atmospheric pressure = " + str(pres) + "h
37
38         #showing the notification
39         notification.notify(title = "YOUR WEATHER REPORT",message=info,timeout=2)
40         # waiting time
```

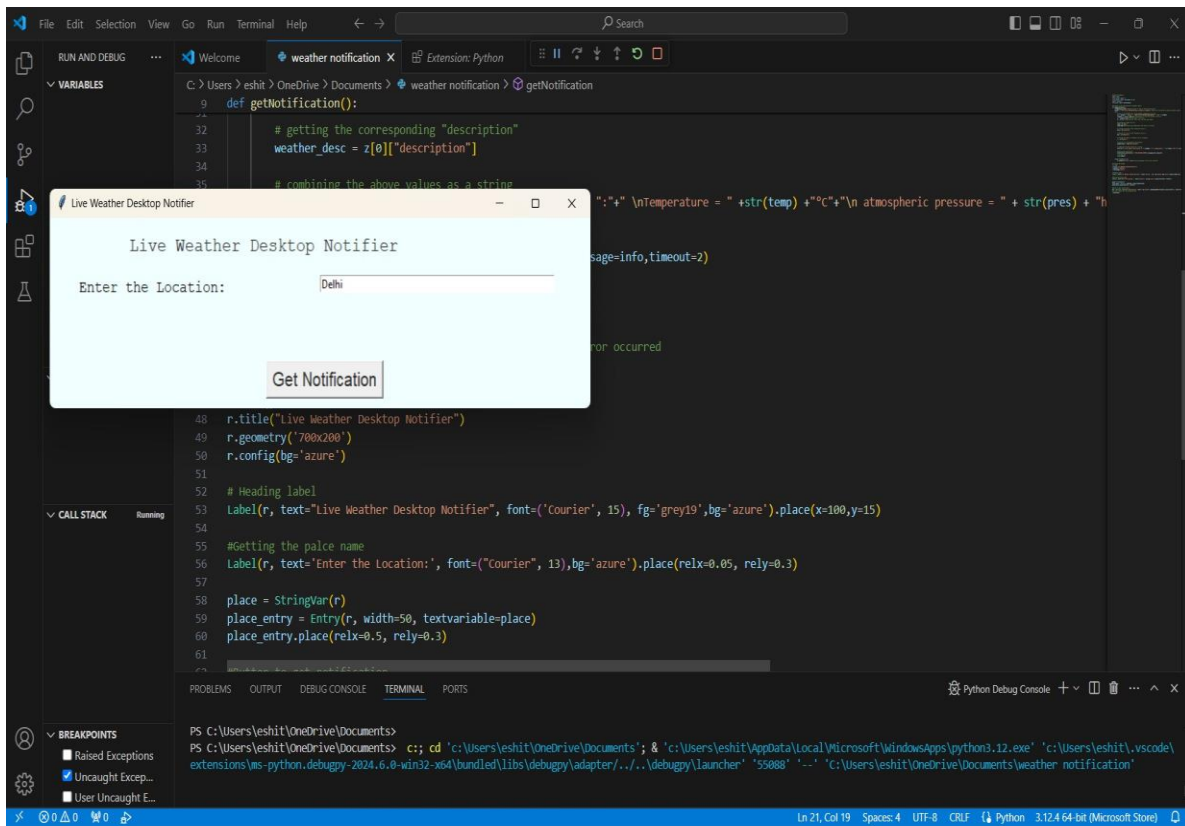
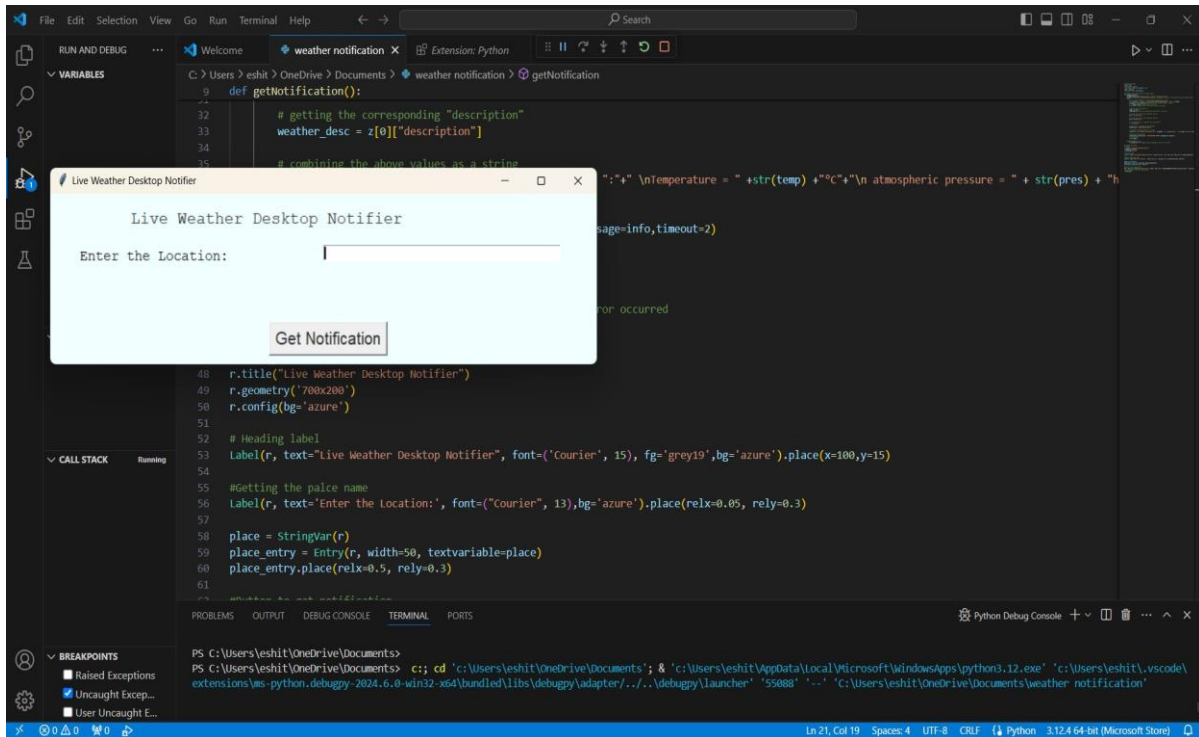
```
Go Run Terminal Help ← → Search
Welcome weather notification Extension: Python
C:\Users\eshit> OneDrive\Documents\weather notification\getNotification.py
9 def getNotification():
13     # This is the complete url to get weather conditions of a city
14     url = baseUrl + "appid=" + 'd850f7f52bf19300a9eb4b0aa6b80f0d' + "&q=" + cityName
15     response = requests.get(url) #requesting for the the content of the url
16     x = response.json() #converting it into json
17     y = x["main"] #getting the "main" key from the json object
18
19     # getting the "temp" key of y
20     temp = y["temp"]
21     temp=temp-273 #converting temperature from kelvin to celcius
22
23     # storing the value of the "pressure" key of y
24     pres = y["pressure"]
25
26     # getting the value of the "humidity" key of y
27     hum = y["humidity"]
28
29     # storing the value of "weather" key in variable z
30     z = x["weather"]
31
32     # getting the corresponding "description"
33     weather_desc = z[0]["description"]
34
35     # combining the above values as a string
36     info="Here is the weather description of "+ cityName+ ":\n" \
37     "Temperature = " +str(temp) + "°C"+ "\n atmospheric pressure = " + str(pres) + "h
38
39     #showing the notification
40     notification.notify(title = "YOUR WEATHER REPORT",message=info,timeout=2)
41     # waiting time
42     time.sleep(7)
43
44     except Exception as e:
45         mb.showerror('Error',e)#show pop up message if any error occurred
46
47 #creating the window
48 r = Tk()
49 r.title("Live Weather Desktop Notifier")
50 r.geometry('700x200')
51 r.config(bg='azure')
```

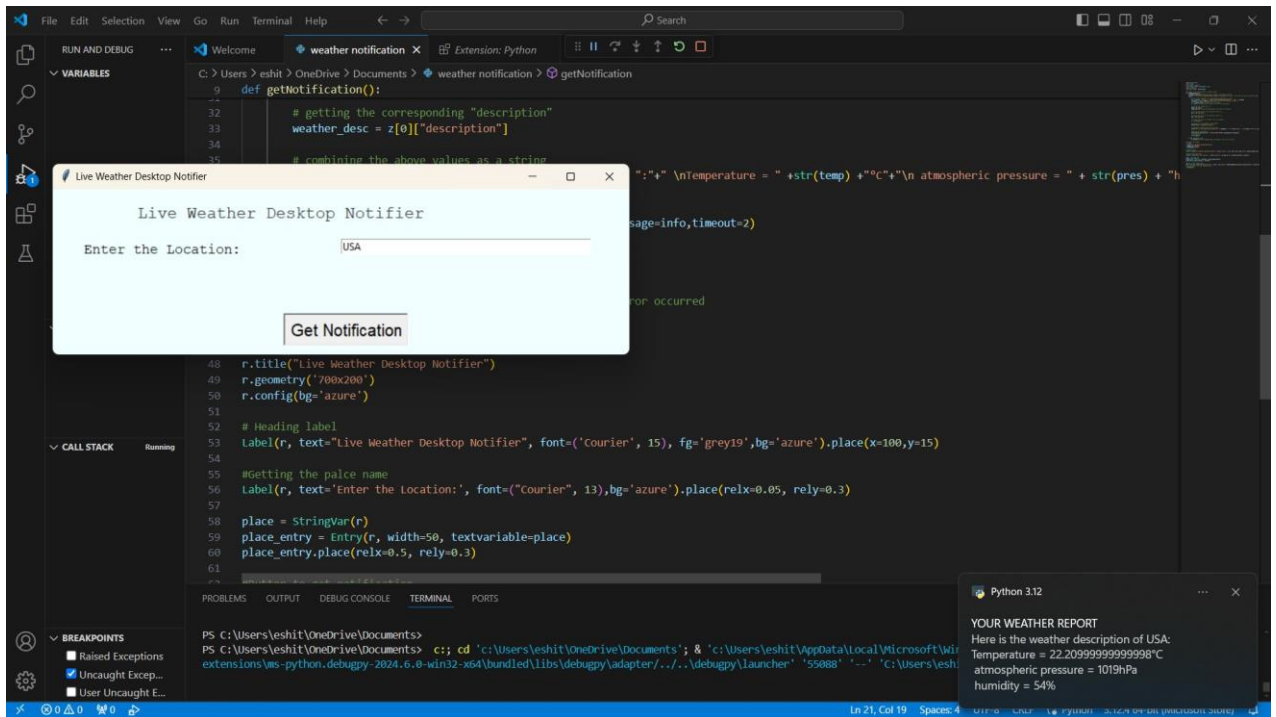
Ln 21, Col 19 Spaces: 4 UTF-8 CRLF Python 3.12.4 64-bit (Microsoft Store)

```
46 #creating the window
47 r = Tk()
48 r.title("Live Weather Desktop Notifier")
49 r.geometry('700x200')
50 r.config(bg='azure')
51
52 # Heading label
53 Label(r, text="Live Weather Desktop Notifier", font=('Courier', 15), fg='grey19',bg='azure').place(x=100,y=15)
54
55 #Getting the palce name
56 Label(r, text='Enter the Location:', font=("Courier", 13),bg='azure').place(relx=0.05, rely=0.3)
57
58 place = StringVar(r)
59 place_entry = Entry(r, width=50, textvariable=place)
60 place_entry.place(relx=0.5, rely=0.3)
61
62 #Button to get notification
63 btn = Button(r, text='Get Notification', font=7, fg='grey19',command=getNotification).place(relx=0.4, rely=0.75)
64 #run the window till the closed by user
65 r.mainloop()
66
```

Ln 21, Col 19 Spaces: 4 UTF-8 CRLF Python 3.12.4 64-bit (Microsoft Store)

OUTPUTS





FUTURE SCOPE OF THE PROJECT

The future aspects of your Live Weather Desktop Notifier project are promising and diverse, offering a wide range of possibilities for growth and development. You can expand its reach by porting it to mobile devices or web applications and integrating it with smart home systems to control temperature, lighting, or security settings based on weather conditions. Advanced weather analytics can be incorporated using machine learning algorithms to predict weather patterns and personalization options can be added to customize notifications, temperature units, and weather data sources. Gamification elements can be introduced to encourage users to stay up-to-date with weather forecasts, and collaboration with weather services can provide premium features or more accurate data. Furthermore, IoT integration can enable automated actions based on weather conditions, such as adjusting thermostat settings or triggering weather-resistant outdoor devices. An enhanced user interface can improve the overall user experience, making it more visually appealing and user-friendly. With the possibility of global weather monitoring, commercialization, and licensing to companies, your project has vast potential for impact and success. By exploring these future aspects, you can continue innovating and improving your project, making it an essential tool for weather enthusiasts and smart home users.

CONCLUSION

In conclusion, this project successfully created a Live Weather Desktop Notifier that provides users with accurate and up-to-date weather information, achieving all the project's objectives. The development process involved designing a user-friendly interface, integrating with reliable weather APIs, and implementing efficient notification systems. Despite encountering challenges in data parsing and UI design, the project was completed successfully, demonstrating my ability to overcome obstacles and deliver a functional product.

Through this project, I developed valuable Python programming skills, enhancing my software development proficiency. The Live Weather Desktop Notifier has the potential to be widely adopted and provide significant value to users, especially those who rely on accurate weather forecasts for their daily activities.

Future developments include expanding the application to mobile devices, integrating with smart home systems, and incorporating additional features such as weather forecasting and alert systems. I recommend this project to anyone interested in developing a practical and useful application that can positively impact people's lives. Overall, this project was a rewarding experience that allowed me to apply my skills and knowledge to create a meaningful product, and I am proud of the outcome.

BIBLIOGRAPHY

- <https://www.geeksforgeeks.org>
- <https://www.w3schools.com>
- <https://www.programiz.com>
- <https://pythongeeks.org>

THANK YOU