

 [microsoft](#) / [aed-wonder-woman](#)generated from [microsoft/aed-learn-template](#)

<> Code

! Issues

🔗 Pull requests

▶ Actions

📁 Projects

📖 Wiki

🛡 Security

 main ▼

...

[aed-wonder-woman](#) / [0-wonderwomanlp](#) / [2-secretmessage](#) / [includes](#) / [2-python-basics-for-decoder.md](#)

sguthals images are in

🕒 History

 1 contributor

Raw

Blame



81 lines (53 sloc) 4.08 KB

Python Basics Needed for a Wonder Woman Message Decoder

Before you can uncover the true meaning of the note, let's walk you through the basics for commanding the Python. If you already know how to use comments, variables, functions, and conditionals, you can skip to the next unit in this module.

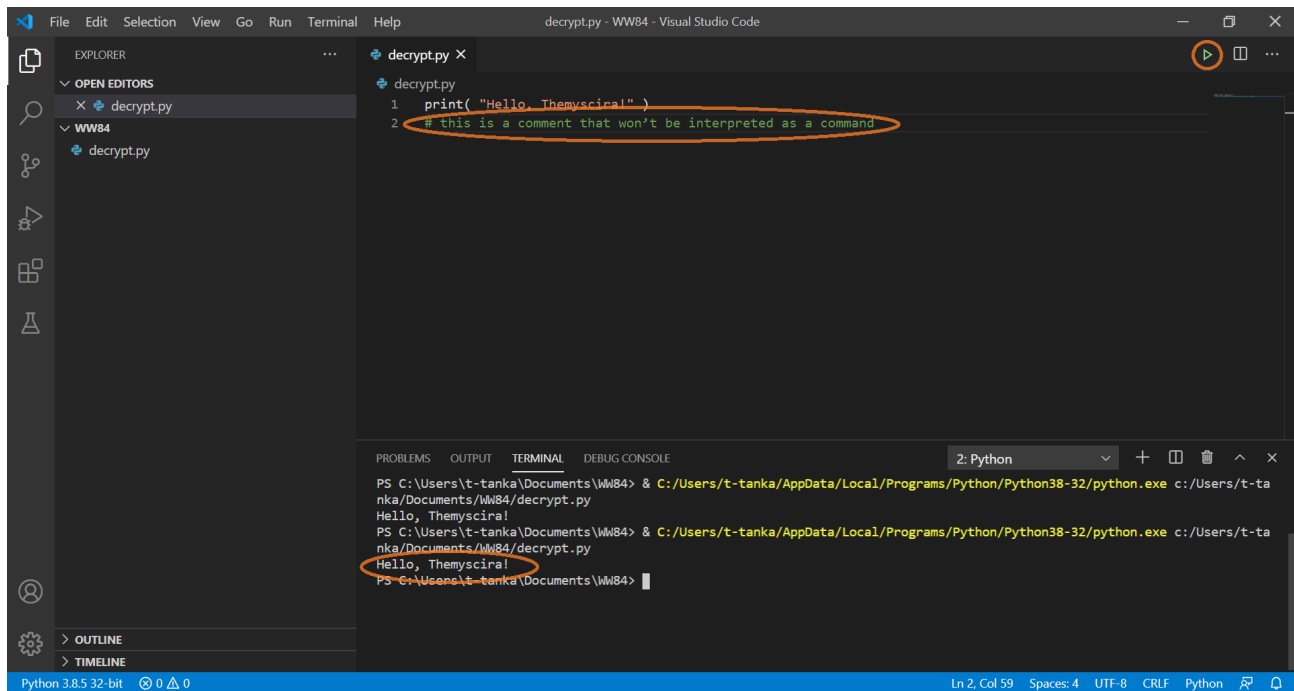
Comments

First, we can write comments to ourselves that the Python will ignore. On any line where we use the `#` symbol, the Python ignores everything after.

Try adding the following to your file:

```
# this is a comment that won't be interpreted as a command
```

Press the Play Button, and you should see the same behavior as before.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a file named `decrypt.py` under a workspace named `WW84`. The main editor shows the contents of `decrypt.py`:

```
1 print( "Hello, Themyscira!" )
2 # this is a comment that won't be interpreted as a command
```

The comment on line 2 is circled in orange. Below the editor, the TERMINAL pane shows the command prompt output:

```
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-tanka/
Documents/WW84/decrypt.py
Hello, Themyscira!
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-ta
nka/Documents/WW84/decrypt.py
Hello, Themyscira!
PS C:\Users\t-tanka\Documents\WW84>
```

The output `Hello, Themyscira!` is circled in orange. The status bar at the bottom indicates `Python 3.8.5 32-bit` and `Ln 2, Col 59`.

Variables

Coding is essentially the movement and manipulation of data; but imagine writing a message decoder on a calculator where the only data you had access to is the last result of an equation. While not impossible, that just seems unnecessarily difficult!

In Python we have "variables", which are like containers that have a name and store data. You can also think of variables like codenames to information that we want our program to remember and have access to later.

The following command tells the Python to associate the codename/variable `diana` with the information "WONDER WOMAN 1984". To do this, you use the assignment `=` sign operator. Any time after this command, when the Python sees variable `diana`, it will substitute in "WONDER WOMAN 1984"!

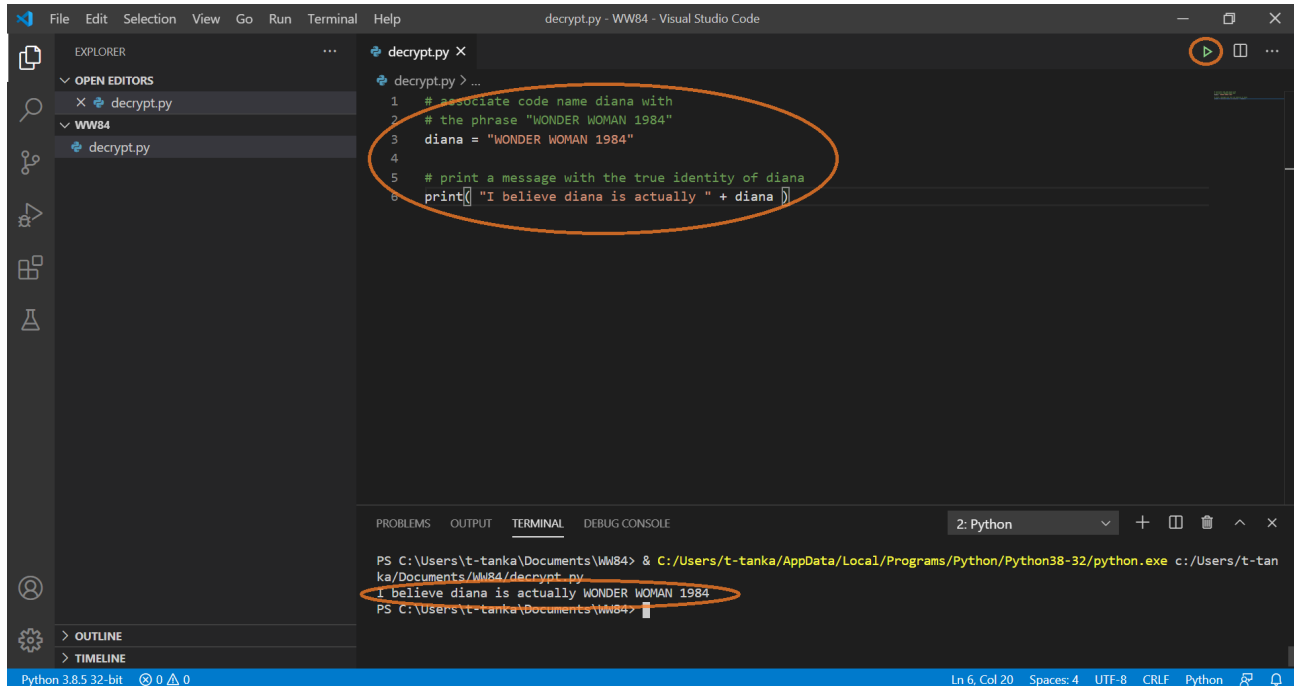
```
# associate variable diana with
# the value "WONDER WOMAN 1984"
diana = "WONDER WOMAN 1984"
```

Try it out by replacing the commands in your file with the following. (If you're wondering what the `+` sign is doing, it "glues together" phrases into a longer phrase.)

```
# associate variable diana with
# the phrase "WONDER WOMAN 1984"
diana = "WONDER WOMAN 1984"
```

```
# print a message with the true identity of diana
print( "I believe diana is actually " + diana )
```

Press the Play Button, and you should see Diana's identity revealed.



Functions

Like variables are containers for single bits of information, functions are like containers for lines of code. Typically, programs run from top to bottom, running each line of code in order. However, if you write a function you can invoke it with by calling it's name whenever and how ever many times you want! Invoking a function means to tell the program to run the code within that function, even if it's in a different part of the file.

Furthermore, you can even add data to the functions as parameters. Parameters are variables that contain data that you want to use to speify the behavior of the function.

Here's how we can give the Python the power to chant. Given a phrase, it can chant it three times.

```
# define a power (function) to chant a phrase
def chant( phrase ):
    # glue three copies together and message it
    print( phrase + phrase + phrase )
```

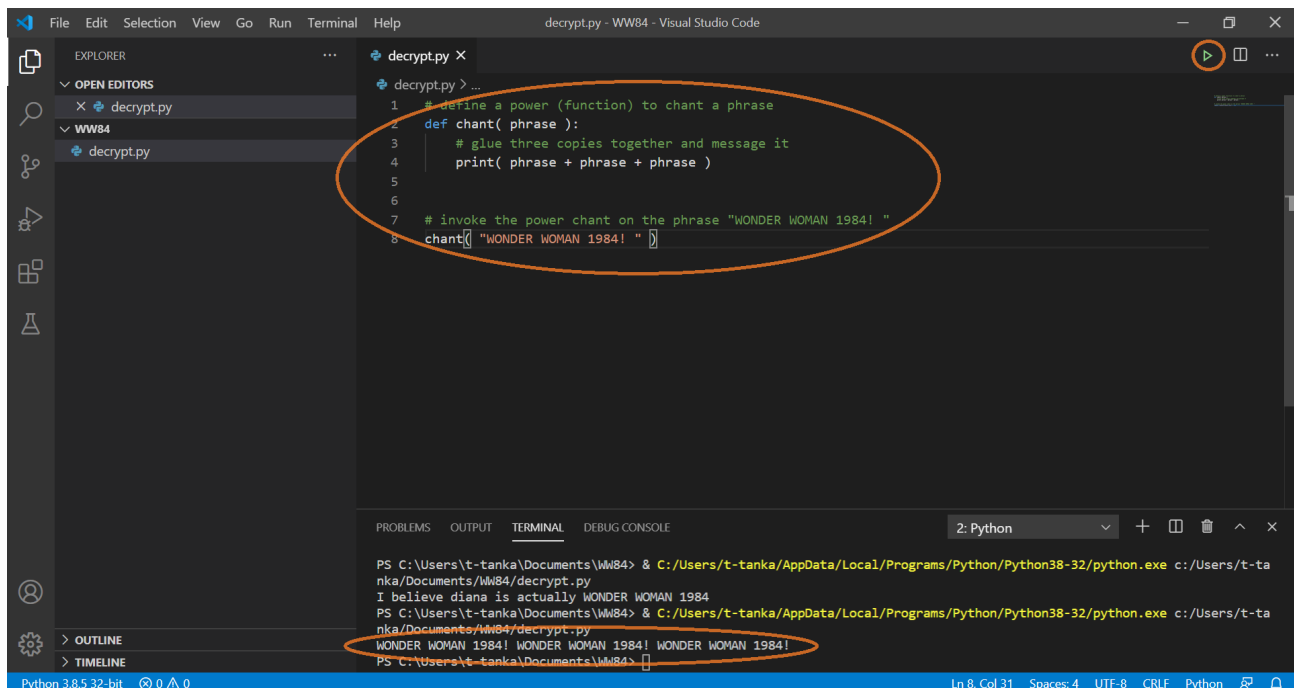
In this code, the function is called `chant` , the parameter is a variable called `phrase` , and the code inside the function invokes a function that you didn't write, but that is available to you as part of the Python language `print()` . If you were to invoke this function, the value you pass in as the parameter will be printed to the console 3 times.

Try it out by replacing the commands in your file with the following to give the power and test it out.

```
# define a power (function) to chant a phrase
def chant( phrase ):
    # glue three copies together and message it
    print( phrase + phrase + phrase )

# invoke the power chant on the phrase "WONDER WOMAN 1984! "
chant( "WONDER WOMAN 1984! " )
```

Press the Play Button, and you should see the chant printed in the console at the bottom of the Visual Studio Code window.



Great! Now that you know the basics, let's get to decrypting the note!

WONDER WOMAN 1984 TM & © DC and WBEI. RATED PG-13