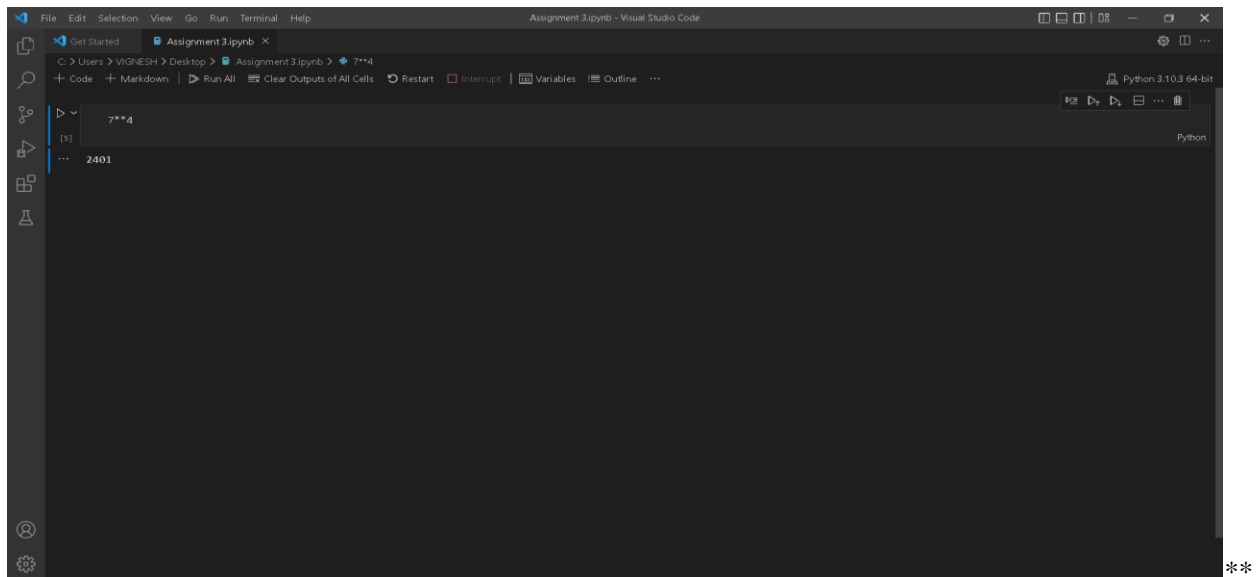


VIGNESH S
211719104159
IBM NALAIYATHIRAN
ASSIGNMENT - 3

Assignment Date	3 October 2022
Student Name	VIGNESH S
Student Roll Number	211719104159
Maximum Marks	2 Marks

<https://colab.research.google.com/drive/1uUeGiokHeIz9YQB9QtIEryC3Lldm60Me?usp=sharing>

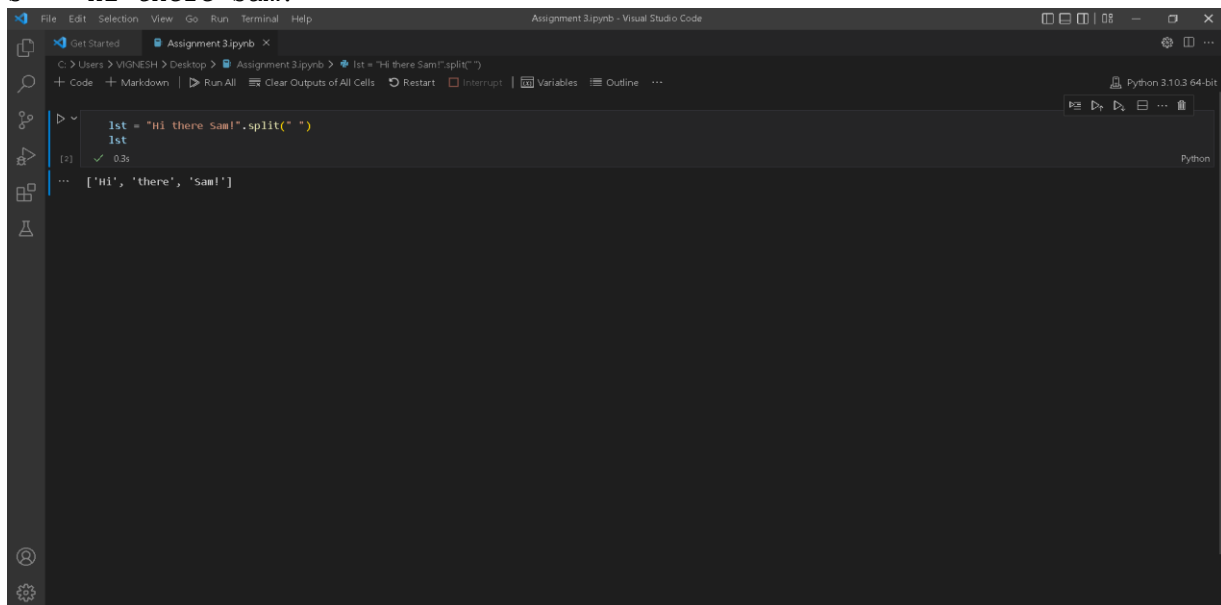
**** What is 7 to the power of 4?****



The screenshot shows a Jupyter Notebook cell in Visual Studio Code. The code entered is `7**4`. The output, displayed below the code, is `2401`. The interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a toolbar with icons for running and clearing cells, and a sidebar with icons for Explorer, Search, and Run and Debug. The bottom status bar indicates the Python version is 3.10.3 64-bit.

****Split this string:****

`s = "Hi there Sam!"`



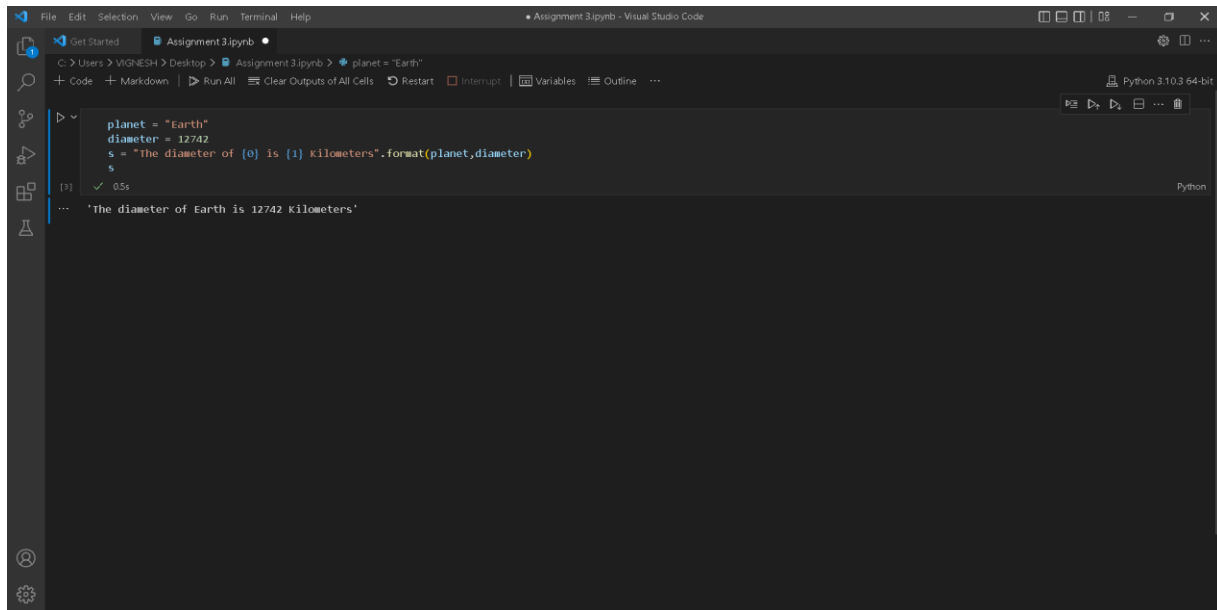
The screenshot shows a Jupyter Notebook cell in Visual Studio Code. The code entered is `s = "Hi there Sam!"` followed by `lst = s.split(" ")`. The output, displayed below the code, is `['Hi', 'there', 'Sam!']`. The interface is similar to the previous screenshot, showing the same menu bar, toolbar, and sidebar. The bottom status bar indicates the Python version is 3.10.3 64-bit.

**** Given the variables:****

```
planet = "Earth"  
diameter = 12742
```

**** Use .format() to print the following string: ****

The diameter of Earth is 12742 kilometers.

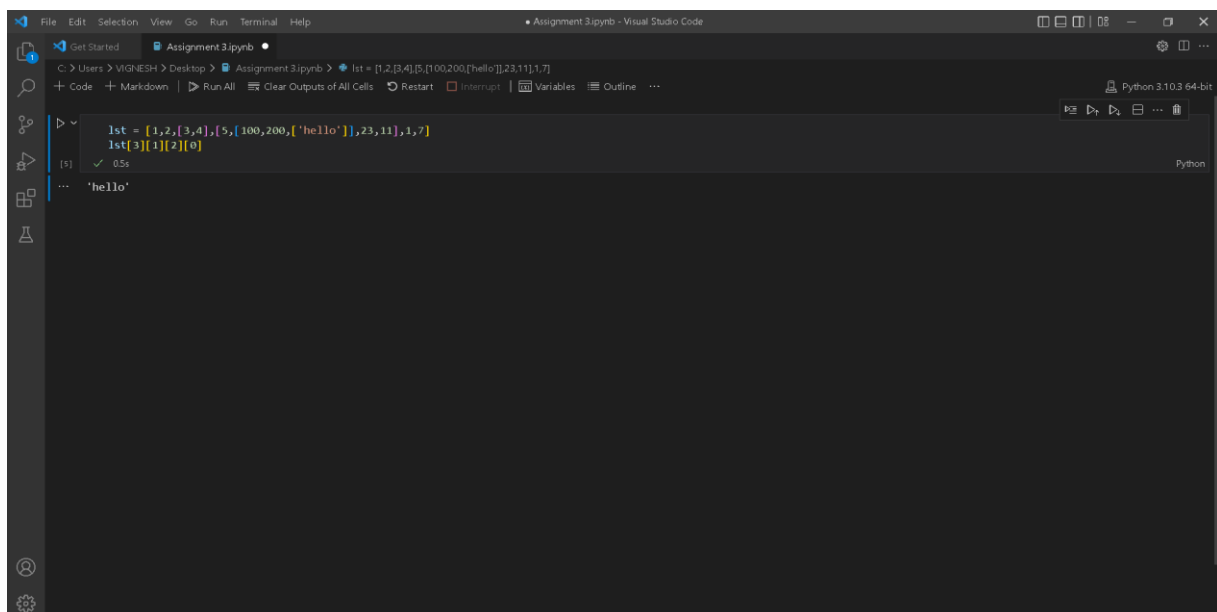


The screenshot shows a Jupyter Notebook interface within Visual Studio Code. The notebook is titled 'Assignment 3.ipynb'. The code cell contains the following Python code:

```
planet = "Earth"  
diameter = 12742  
s = "The diameter of {} is {} kilometers".format(planet,diameter)  
s
```

The output of the cell is displayed below the code, showing the string: "The diameter of Earth is 12742 kilometers". The execution time is 0.5s.

**** Given this nested list, use indexing to grab the word "hello" ****

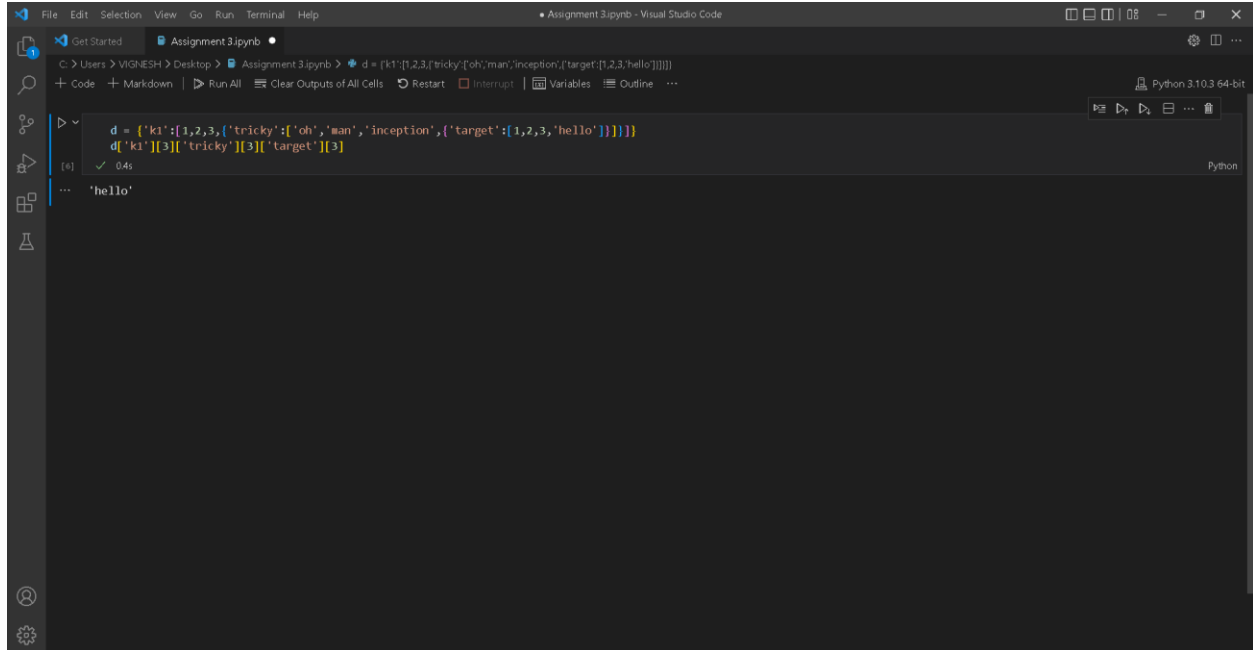


The screenshot shows a Jupyter Notebook interface within Visual Studio Code. The notebook is titled 'Assignment 3.ipynb'. The code cell contains the following Python code:

```
lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]  
lst[3][1][2][0]
```

The output of the cell is displayed below the code, showing the string: "hello". The execution time is 0.5s.

**** Given this nest dictionary grab the word "hello". Be prepared, this will be annoying/tricky ****



The screenshot shows a Jupyter Notebook cell in Visual Studio Code. The code defines a nested dictionary `d` and prints its value at a specific index. The dictionary is structured as follows: `d = {'k1': [1, 2, 3, {'tricky': ['oh', 'man', 'inception', {'target': [1, 2, 3, 'hello']}]}]}`. The output of the cell is `'hello'`.

```
d = {'k1': [1, 2, 3, {'tricky': ['oh', 'man', 'inception', {'target': [1, 2, 3, 'hello']}]}]}
d['k1'][3]['tricky'][3]['target'][3]
```

```
'hello'
```

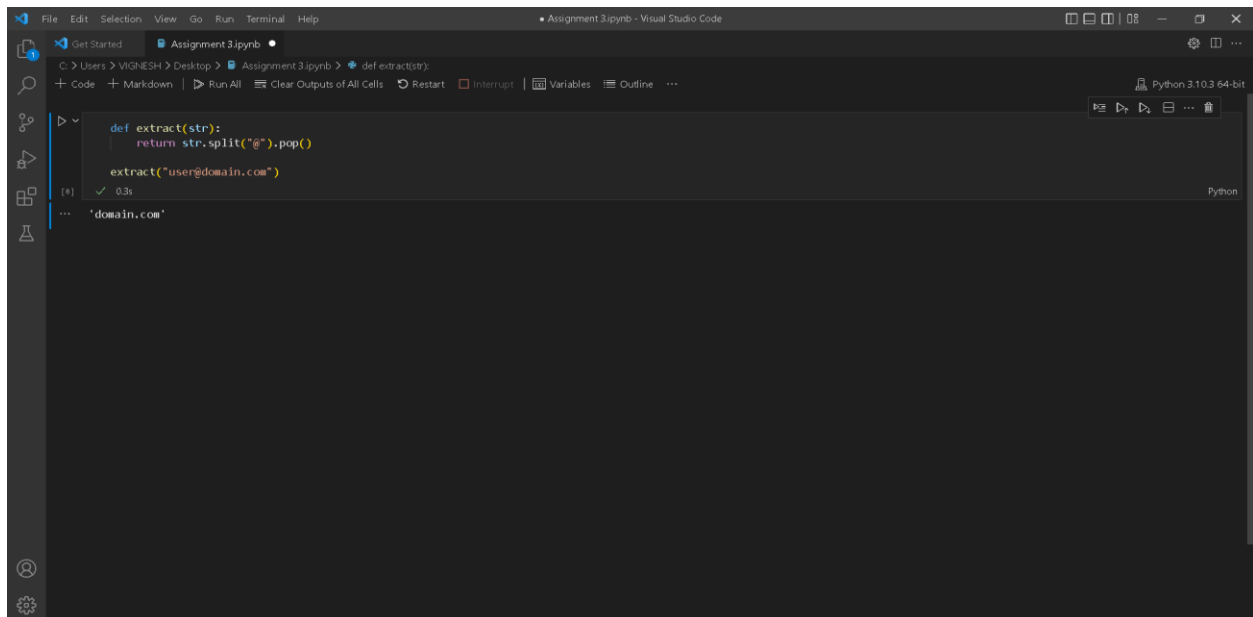
**** What is the main difference between a tuple and a list? ****

#The main difference is list are mutable whereas tuples are immutable. List are defined using `[]` and tuples are defined using `()`

**** Create a function that grabs the email website domain from a string in the form: ****

user@domain.com

So, for example, passing "user@domain.com" would return: domain.com



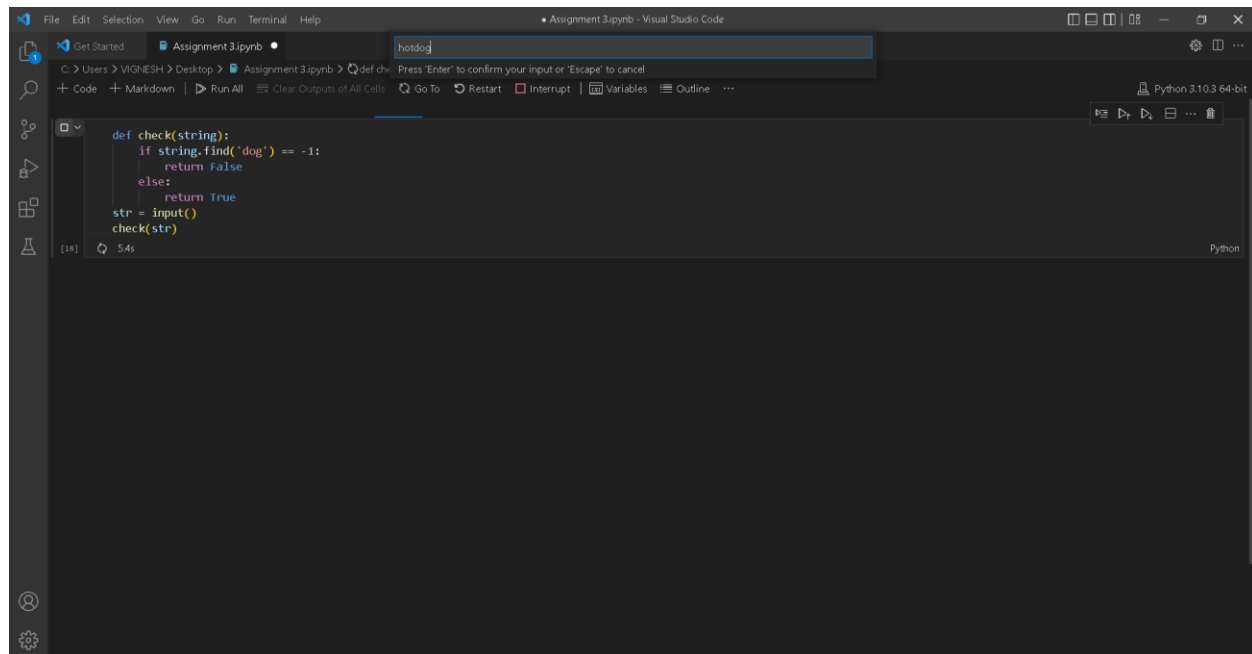
The screenshot shows a Jupyter Notebook cell in Visual Studio Code. The code defines a function `extract` that takes a string and returns the domain part of an email address. The function is defined as `def extract(str): return str.split("@").pop()`. The output of the cell is `'domain.com'`.

```
def extract(str):
    return str.split("@").pop()

extract("user@domain.com")
```

```
'domain.com'
```

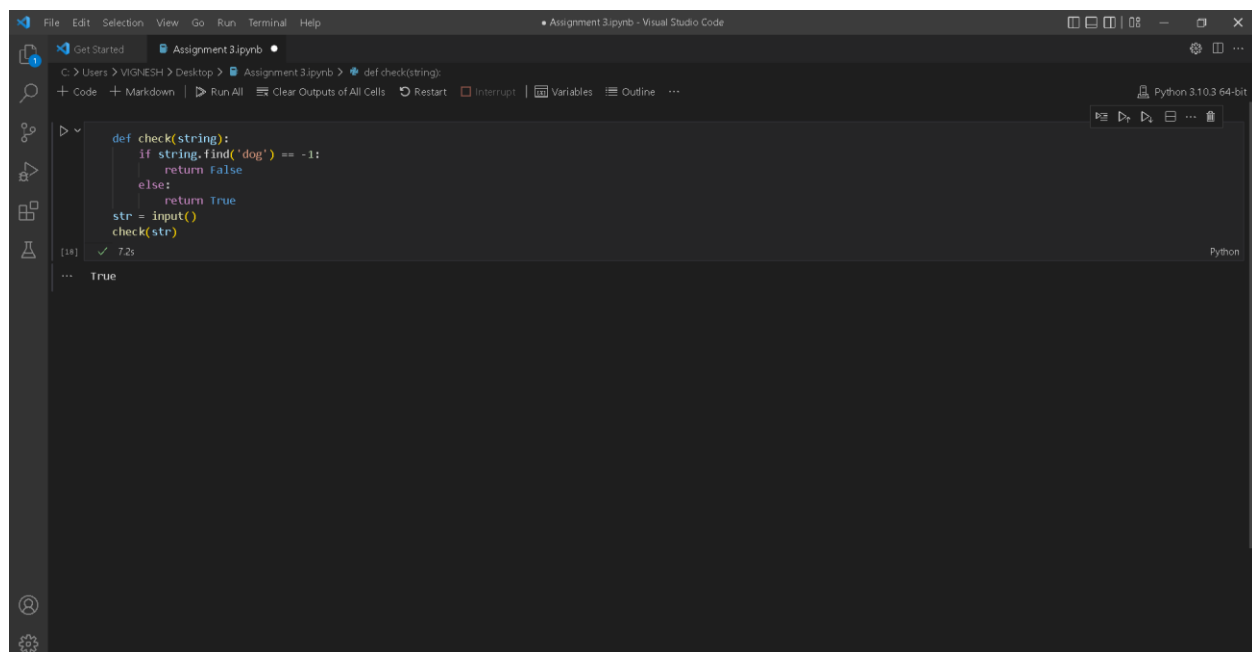
**** Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization. ****



The screenshot shows the Visual Studio Code interface with a Jupyter Notebook. The code editor displays the following Python code:

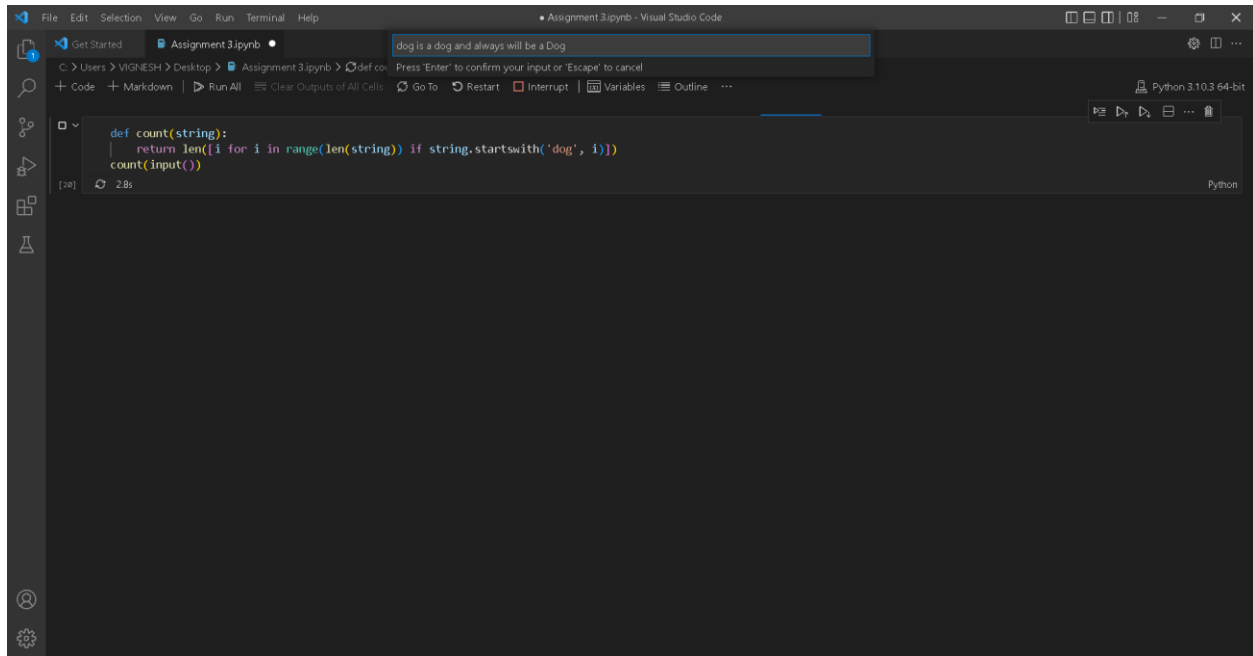
```
def check(string):  
    if string.find('dog') == -1:  
        return False  
    else:  
        return True  
str = input()  
check(str)
```

The Jupyter Notebook interface shows a prompt "holddog" in the input field, indicating that the user has entered the word "holddog" as input for the function.



The screenshot shows the same Visual Studio Code interface with the Jupyter Notebook. The code is the same as in the previous screenshot. The Jupyter Notebook interface now shows the output of the function, which is "True", indicating that the word "dog" was found in the input string "holddog".

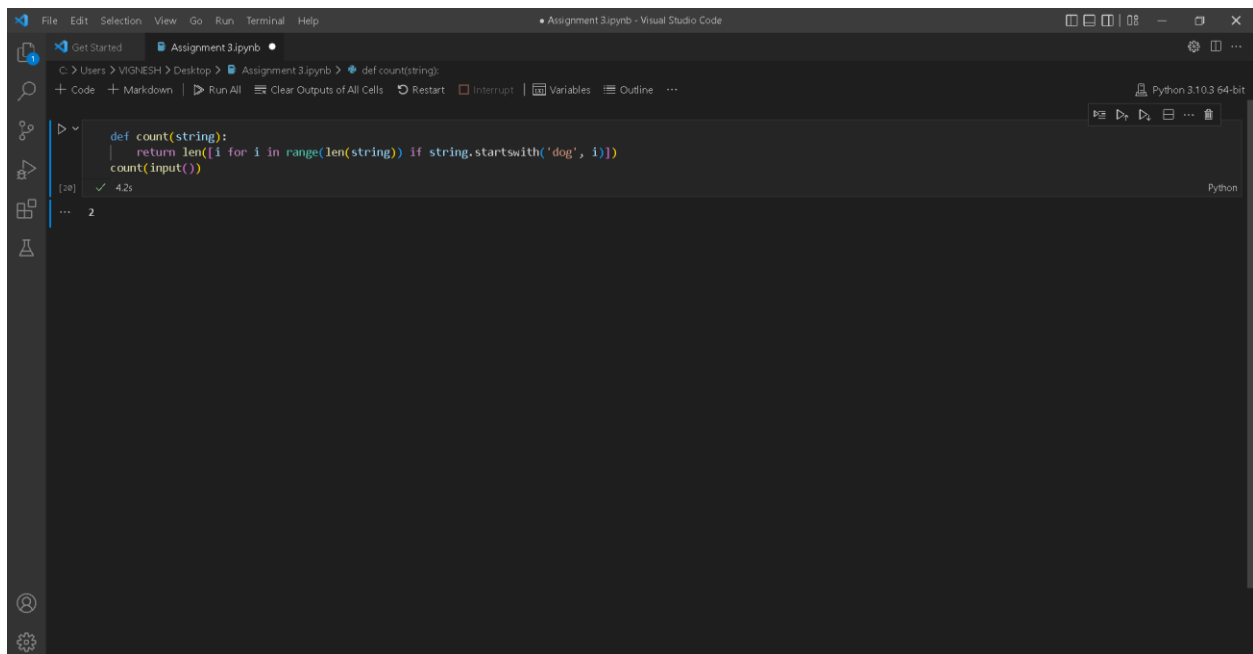
**** Create a function that counts the number of times the word "dog" occurs in a string. Again ignore edge cases. ****



The screenshot shows a Jupyter Notebook cell in Visual Studio Code. The cell contains the following Python code:

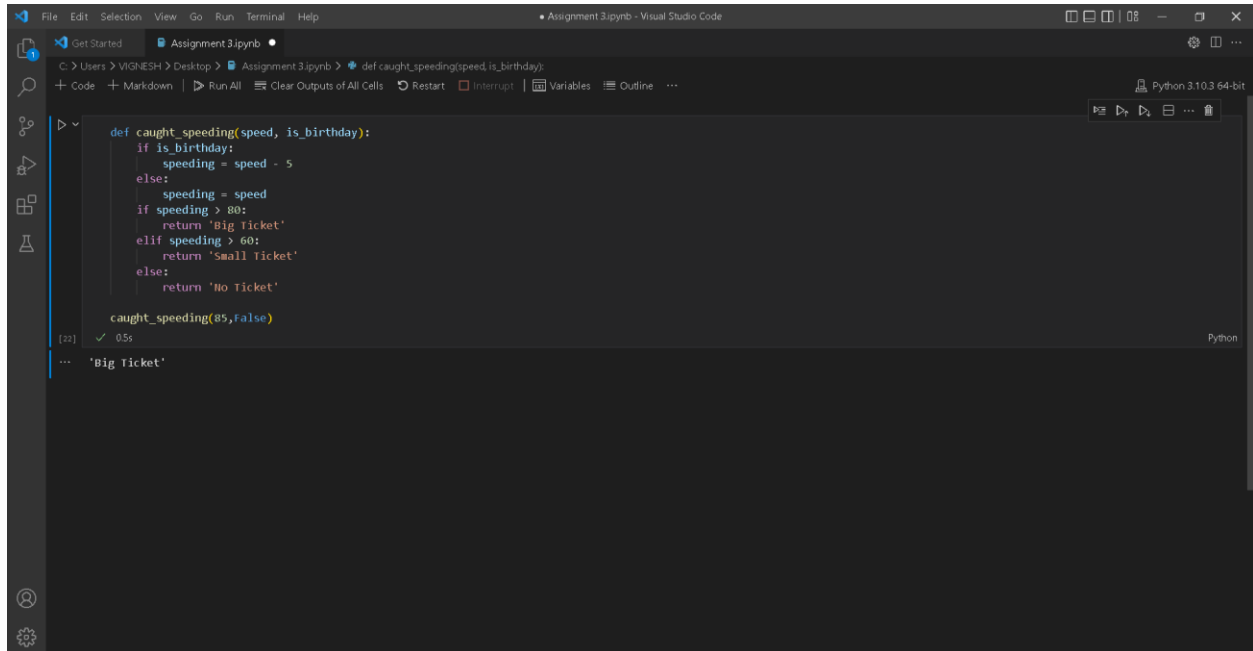
```
def count(string):  
    return len([i for i in range(len(string)) if string.startswith('dog', i)])  
count(input())
```

The input prompt is "dog is a dog and always will be a Dog". The output of the cell is "2.8s", indicating the execution time.



The screenshot shows the same Jupyter Notebook cell in Visual Studio Code, but now with the output of the function. The output is "4.2s", indicating the execution time. Below the output, the number "2" is displayed, which is the result of the function call.

You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: "No ticket", "Small ticket", or "Big Ticket". If your speed is 60 or less, the result is "No Ticket". If speed is between 61 and 80 inclusive, the result is "Small Ticket". If speed is 81 or more, the result is "Big Ticket". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.

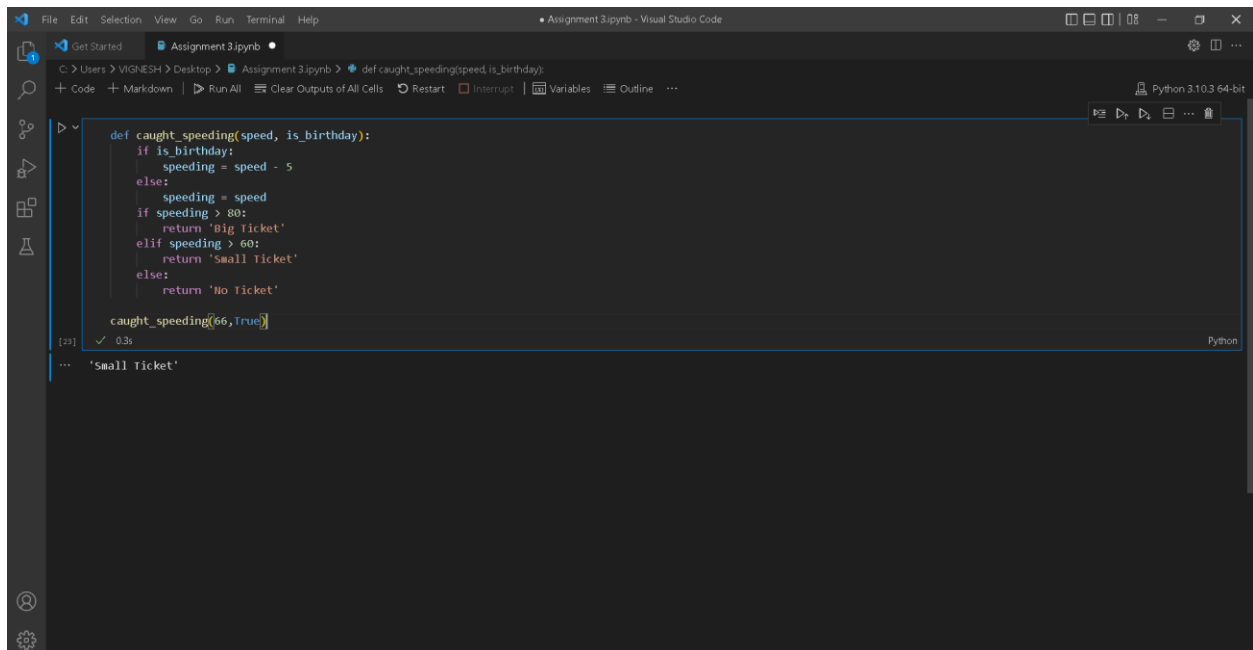


```
def caught_speeding(speed, is_birthday):
    if is_birthday:
        speeding = speed - 5
    else:
        speeding = speed
    if speeding > 80:
        return 'Big Ticket'
    elif speeding > 60:
        return 'Small Ticket'
    else:
        return 'No Ticket'

caught_speeding(85, False)
```

[22] ✓ 0.5s Python

... 'Big Ticket'



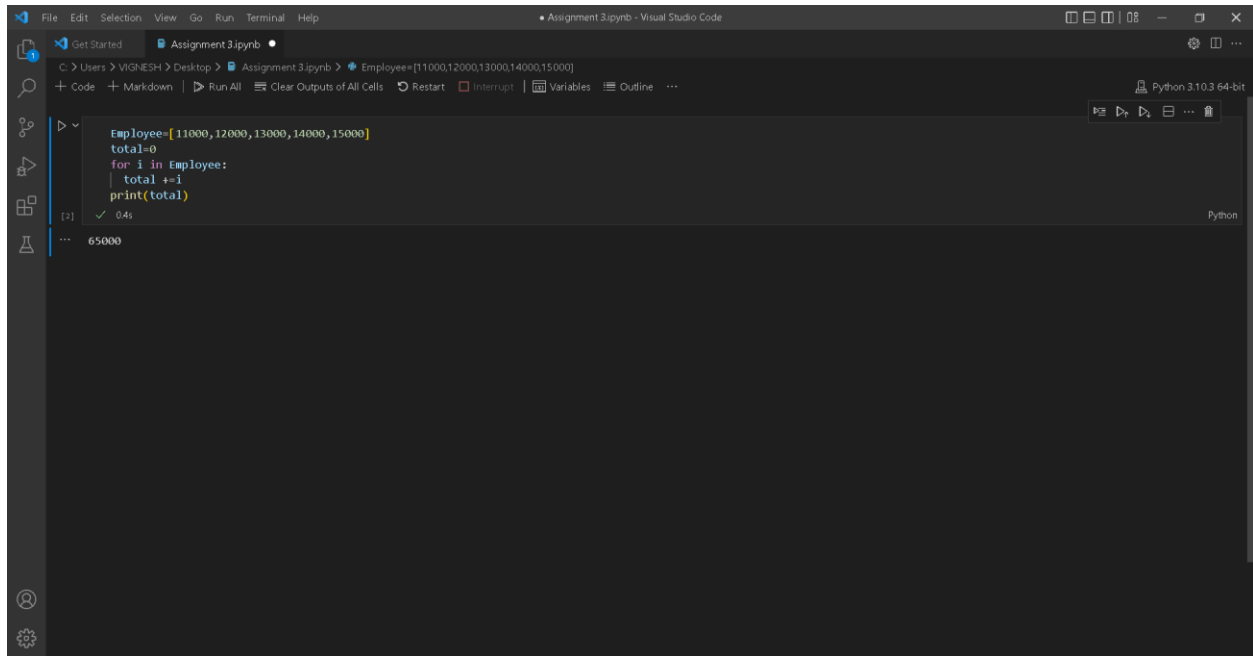
```
def caught_speeding(speed, is_birthday):
    if is_birthday:
        speeding = speed - 5
    else:
        speeding = speed
    if speeding > 80:
        return 'Big Ticket'
    elif speeding > 60:
        return 'Small Ticket'
    else:
        return 'No Ticket'

caught_speeding(66, True)
```

[23] ✓ 0.3s Python

... 'Small Ticket'

Create an employee list with basic salary values(at least 5 values for 5 employees) and using a for loop retrieve each employee salary and calculate total salary expenditure. Create two dictionaries in Python:



```
File Edit Selection View Go Run Terminal Help
Assignment 3.ipynb - Visual Studio Code
C:\Users\> VIGNESH > Desktop > Assignment 3.ipynb > Employee=[11000,12000,13000,14000,15000]
+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt Variables Outline ...
Python 3.10.3 64-bit

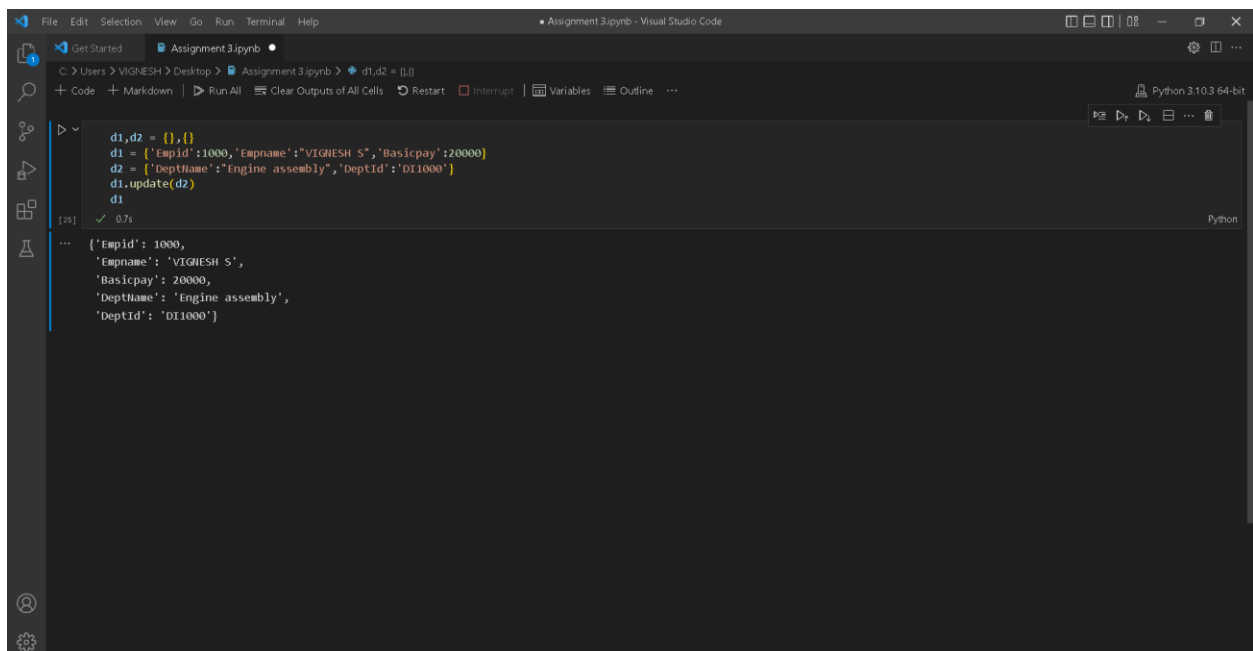
Employee=[11000,12000,13000,14000,15000]
total=0
for i in Employee:
    total+=i
print(total)

[2]: ✓ 0.4s
65000
```

First one to contain fields as Empid, Empname, Basicpay

Second dictionary to contain fields as DeptName, DeptId.

Combine both dictionaries.



```
File Edit Selection View Go Run Terminal Help
Assignment 3.ipynb - Visual Studio Code
C:\Users\> VIGNESH > Desktop > Assignment 3.ipynb > d1,d2 = {},{}
+ Code + Markdown | Run All Clear Outputs of All Cells Restart Interrupt Variables Outline ...
Python 3.10.3 64-bit

d1,d2 = {},{}
d1 = {'Empid':1000,'Empname':'VIGNESH S','Basicpay':20000}
d2 = {'DeptName':'Engine assembly','DeptId':'DI1000'}
d1.update(d2)
d1

[28]: ✓ 0.7s
{'Empid': 1000,
 'Empname': 'VIGNESH S',
 'Basicpay': 20000,
 'DeptName': 'Engine assembly',
 'DeptId': 'DI1000'}
```