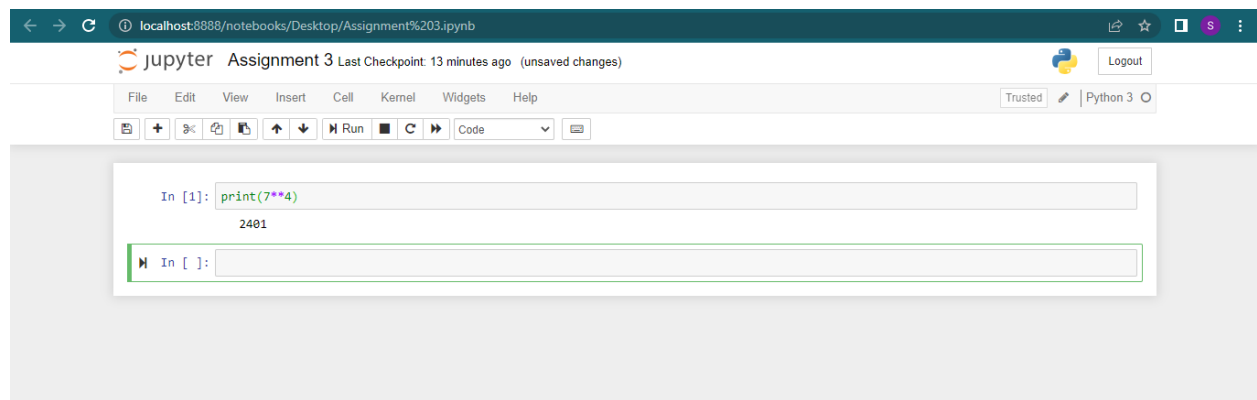


Assignment 3

Answer the questions or complete the tasks outlined in bold below, use the specific method described if applicable.

What is 7 to the power of 4?



A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Desktop/Assignment%203.ipynb'. The notebook title is 'Assignment 3' with a status 'Last Checkpoint: 13 minutes ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations, running, and code execution. The code cell contains the following text:

```
In [1]: print(7**4)
```

The output of the cell is displayed below the code:

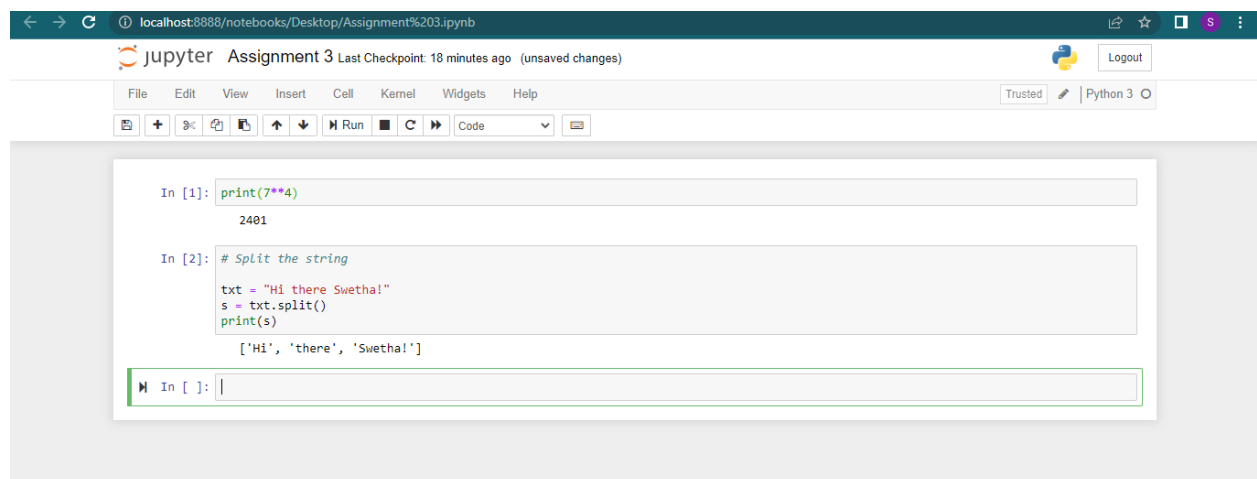
```
2401
```

Below the output, there is an empty code cell with the prompt 'In []:'.

Split this string:

`s = "Hi there Swetha!"`

into a list.



A screenshot of a Jupyter Notebook interface, similar to the one above. The notebook title is 'Assignment 3' with a status 'Last Checkpoint: 18 minutes ago (unsaved changes)'. The code cell contains the following text:

```
In [1]: print(7**4)
```

The output of the cell is displayed below the code:

```
2401
```

The next code cell contains the following text:

```
In [2]: # Split the string
txt = "Hi there Swetha!"
s = txt.split()
print(s)
```

The output of the second cell is displayed below the code:

```
['Hi', 'there', 'Swetha!']
```

Below the output, there is an empty code cell with the prompt 'In []:'.

```
In [1]: print(7**4)
2401

In [2]: # Split the string
txt = "Hi there Swethal!"
s = txt.split()
print(s)

['Hi', 'there', 'Swethal!']

In [3]: txt="Hi this is Sakthi!"
x=txt.split(" ")
print(x)

['Hi', 'this', 'is', 'Sakthi!']

In [ ]:
```

Given the variables:

planet = "Earth" diameter = 12742

Use .format() to print the following string:

The diameter of Earth is 12742 kilometers.

```
In [2]: # Split the string
txt = "Hi there Swethal!"
s = txt.split()
print(s)

['Hi', 'there', 'Swethal!']

In [3]: txt="Hi this is Sakthi!"
x=txt.split(" ")
print(x)

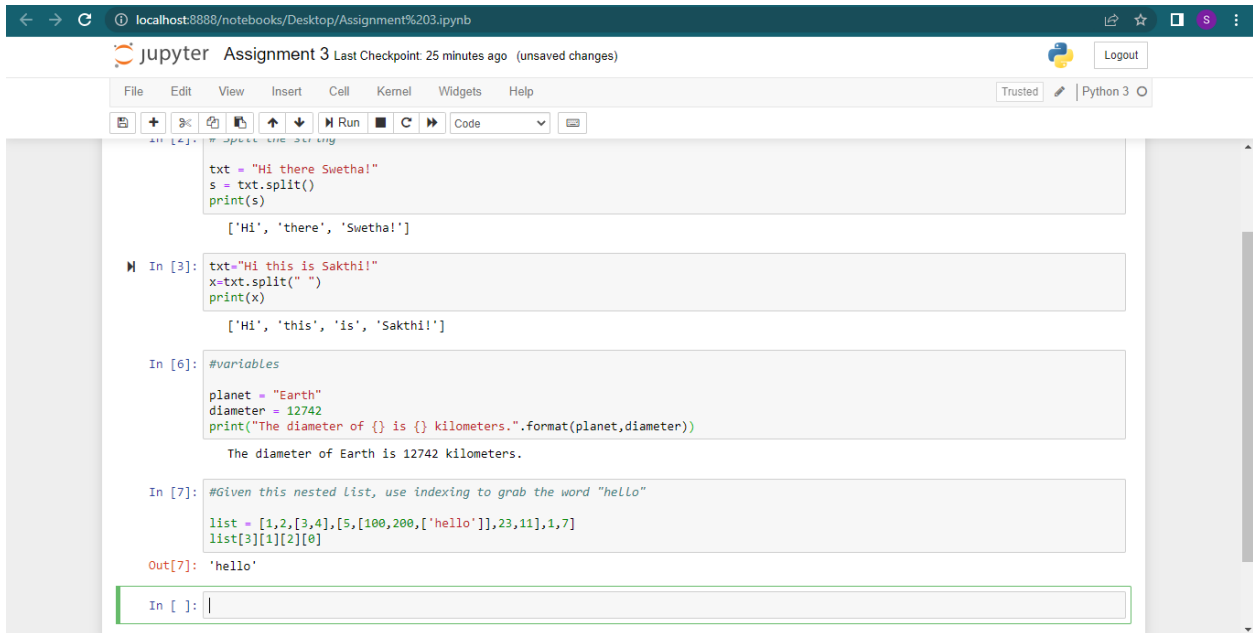
['Hi', 'this', 'is', 'Sakthi!']

In [6]: #variables
planet = "Earth"
diameter = 12742
print("The diameter of {} is {} kilometers.".format(planet,diameter))

The diameter of Earth is 12742 kilometers.

In [ ]:
```

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



A Jupyter Notebook interface titled "Assignment 3" with a last checkpoint 25 minutes ago. The notebook contains several code cells. The first cell shows a string being split into a list. The second cell shows a string with spaces being split into a list. The third cell shows variables for planet and diameter, and a formatted string. The fourth cell shows a nested list and the output of indexing it to retrieve the word "hello".

```
txt = "Hi there Swetha!"
s = txt.split()
print(s)

['Hi', 'there', 'Swetha!']

In [3]: txt="Hi this is Sakthi!"
x=txt.split(" ")
print(x)

['Hi', 'this', 'is', 'Sakthi!']

In [6]: #variables
planet = "Earth"
diameter = 12742
print("The diameter of {} is {} kilometers.".format(planet,diameter))

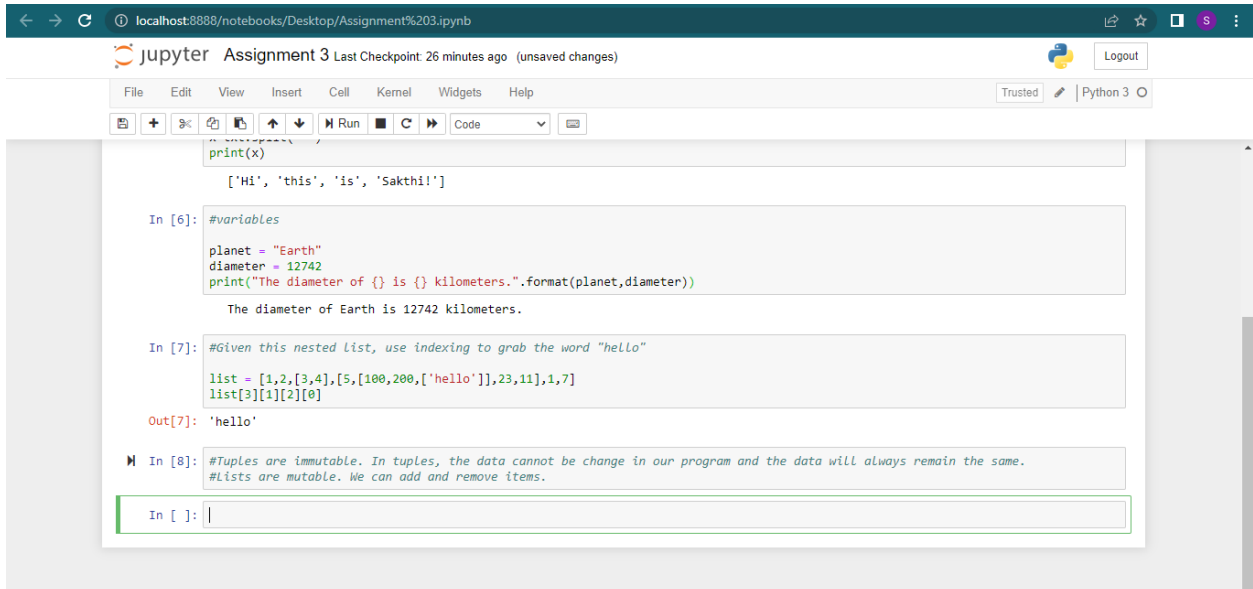
The diameter of Earth is 12742 kilometers.

In [7]: #Given this nested List, use indexing to grab the word "hello"
list = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]
list[3][1][2][0]

Out[7]: 'hello'

In [ ]: 
```

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



A Jupyter Notebook interface titled "Assignment 3" with a last checkpoint 26 minutes ago. The notebook contains several code cells. The first cell shows a string being split into a list. The second cell shows variables for planet and diameter, and a formatted string. The third cell shows a nested list and the output of indexing it to retrieve the word "hello". The fourth cell shows a comment about the difference between tuples and lists.

```
print(x)

['Hi', 'this', 'is', 'Sakthi!']

In [6]: #variables
planet = "Earth"
diameter = 12742
print("The diameter of {} is {} kilometers.".format(planet,diameter))

The diameter of Earth is 12742 kilometers.

In [7]: #Given this nested List, use indexing to grab the word "hello"
list = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]
list[3][1][2][0]

Out[7]: 'hello'

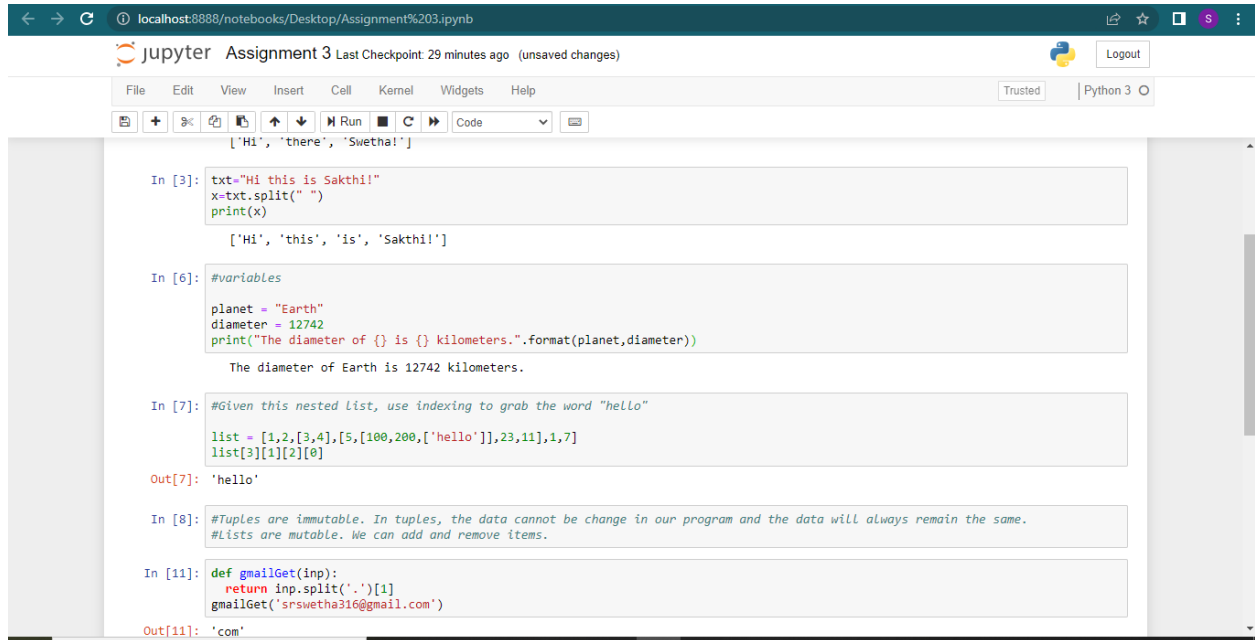
In [8]: #Tuples are immutable. In tuples, the data cannot be change in our program and the data will always remain the same.
#Lists are mutable. We can add and remove items.

In [ ]: 
```

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

srswetha316@gmail.com

So for example, passing " srswetha316@gmail.com" would return: gmail.com



The screenshot shows a Jupyter Notebook titled "Assignment 3" with a toolbar at the top. The notebook contains several code cells:

- Cell 3: A string `txt="Hi this is Sakthi!"` is split by spaces, resulting in the list `['Hi', 'this', 'is', 'Sakthi!']`.
- Cell 6: Variables `planet = "Earth"` and `diameter = 12742` are defined, and a formatted string is printed: `"The diameter of {} is {} kilometers."`, resulting in `The diameter of Earth is 12742 kilometers.`
- Cell 7: A nested list `list = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]` is used to access the word `'hello'` via indexing: `list[3][1][2][0]`.
- Cell 8: A comment explaining that tuples are immutable and lists are mutable.
- Cell 11: A function `gmailGet(inp)` is defined to return the domain part of an email address. It is called with `'srswetha316@gmail.com'`, returning `'com'`.

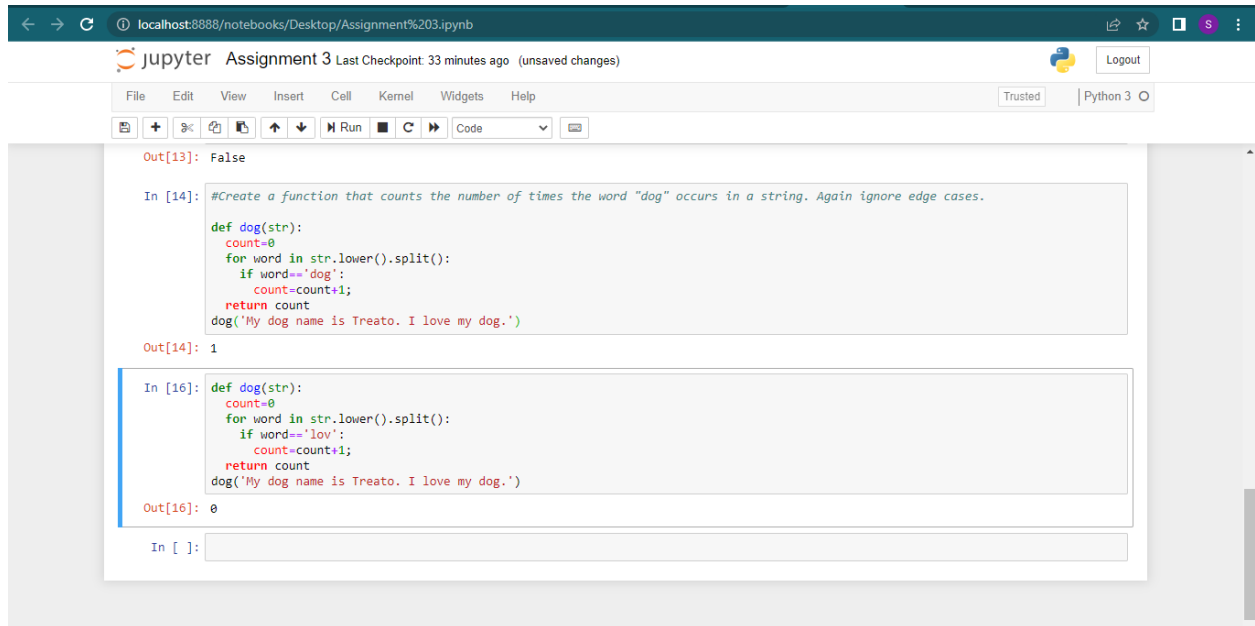
Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization.



The screenshot shows a Jupyter Notebook with the following code cells:

- Cell 12: A function `findDog(inp)` is defined to check if the word `'dog'` is in the input string (case-insensitive). It is called with `'Is there a dog here?'`, returning `True`.
- Cell 13: The same function `findDog` is called with `'Is there a cat here?'`, returning `False`.
- Below cell 13, there is an empty input prompt `In []:`.

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



```
Out[13]: False

In [14]: #Create a function that counts the number of times the word "dog" occurs in a string. Again ignore edge cases.

def dog(str):
    count=0
    for word in str.lower().split():
        if word=="dog":
            count=count+1;
    return count
dog('My dog name is Treato. I love my dog.')
```

```
Out[14]: 1

In [16]: def dog(str):
count=0
for word in str.lower().split():
    if word=="lov":
        count=count+1;
    return count
dog('My dog name is Treato. I love my dog.')
```

```
Out[16]: 0

In [ ]:
```

Problem

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

The image shows a Jupyter Notebook window titled "Assignment 3" with a last checkpoint 38 minutes ago. The notebook is running on a local host at localhost:8888. The code in the notebook defines a function `caught_speeding` that takes `speed` and `is_birthday` as arguments. The function logic is as follows:

```
#Problem
#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 69, the result is "Small Ticket". If speed is 70 or more, the result is "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'

print(caught_speeding(80, True))
```

The output of the first cell is "Big Ticket".

The second cell contains the same function definition but with a different call to `print`:

```
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'

print(caught_speeding(78, True))
```

The output of the second cell is "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.

The image shows a Jupyter Notebook window titled "Assignment 3" with a last checkpoint 38 minutes ago and unsaved changes. The notebook is running on a local host at localhost:8888. The code in the notebook defines a function `caught_speeding` and then creates an employee list and calculates the total salary expenditure.

```
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'

print(caught_speeding(78, True))
```

The output of the first cell is "Small Ticket".

The second cell contains the following code:

```
In [19]: #Create an employee list with basic salary values(at least 5 values for 5 employees) and using a for loop retrieve each employee s

employee = [{"Swetha S", 50000}, {"Sowmi G", 52000}, {"Raji M", 51000}, {"Kiruba R", 53000}, {"Resh S", 55000}]
total = 0
for i in employee:
    print(i[1])
    total += i[1]

print("Total salary expenditure =", total)
```

The output of the second cell is:

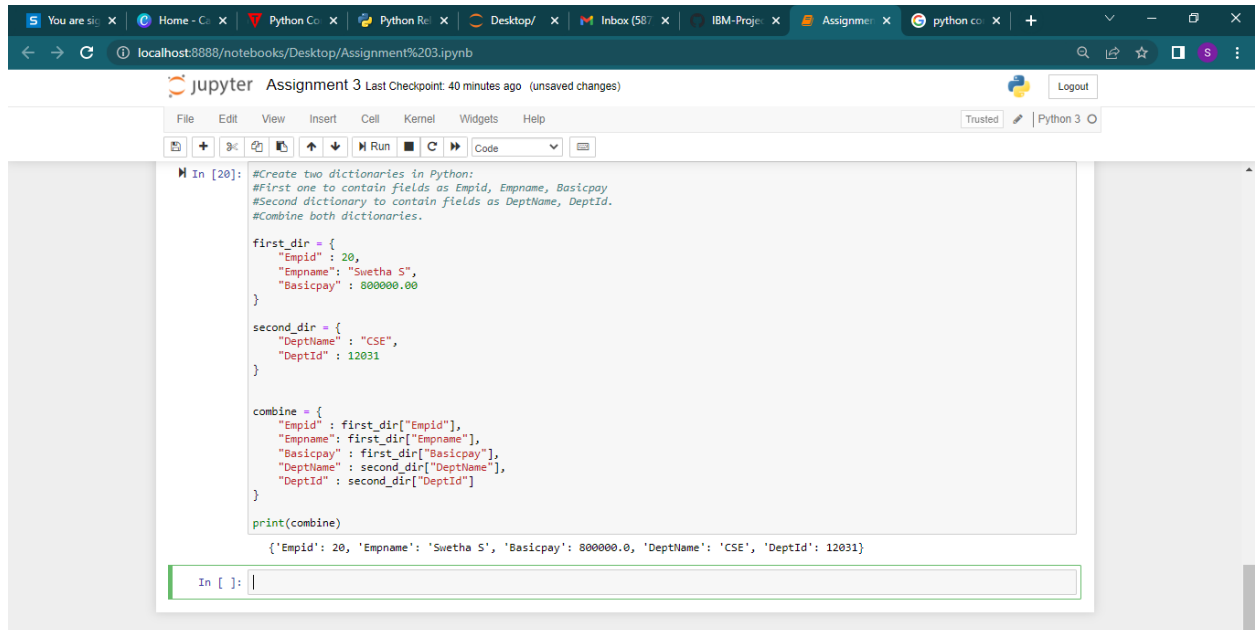
```
50000
52000
51000
53000
55000
Total salary expenditure = 261000
```

Create two dictionaries in Python:

First one to contain fields as Empid, Empname, Basicpay

Second dictionary to contain fields as DeptName, DeptId.

Combine both dictionaries.



The screenshot shows a Jupyter Notebook window titled "Assignment 3 Last Checkpoint: 40 minutes ago (unsaved changes)". The notebook contains a single code cell with the following Python code:

```
#Create two dictionaries in Python:
#First one to contain fields as Empid, Empname, Basicpay
#Second dictionary to contain fields as DeptName, DeptId.
#Combine both dictionaries.

first_dir = {
    "Empid" : 20,
    "Empname" : "Swetha S",
    "Basicpay" : 800000.00
}

second_dir = {
    "DeptName" : "CSE",
    "DeptId" : 12031
}

combine = {
    "Empid" : first_dir["Empid"],
    "Empname" : first_dir["Empname"],
    "Basicpay" : first_dir["Basicpay"],
    "DeptName" : second_dir["DeptName"],
    "DeptId" : second_dir["DeptId"]
}

print(combine)
```

The output of the code is displayed below the cell:

```
{'Empid': 20, 'Empname': 'Swetha S', 'Basicpay': 800000.0, 'DeptName': 'CSE', 'DeptId': 12031}
```