

### ASSIGNMENT - 3

### PYTHON PROGRAMMING

Assignment Date	01 September 2022
Student Name	Sriman Balaji. R
Student Roll Number	1902230
Maximum Marks	2 Marks

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?**

In [1]:

```
7 **4
```

Out[1]:

```
2401
```

**Split this string:** s =

```
"Hi there Sam!"
```

**into a list.**

In [4]:

```
s = 'Hi there Sam!'
```

In [3]:

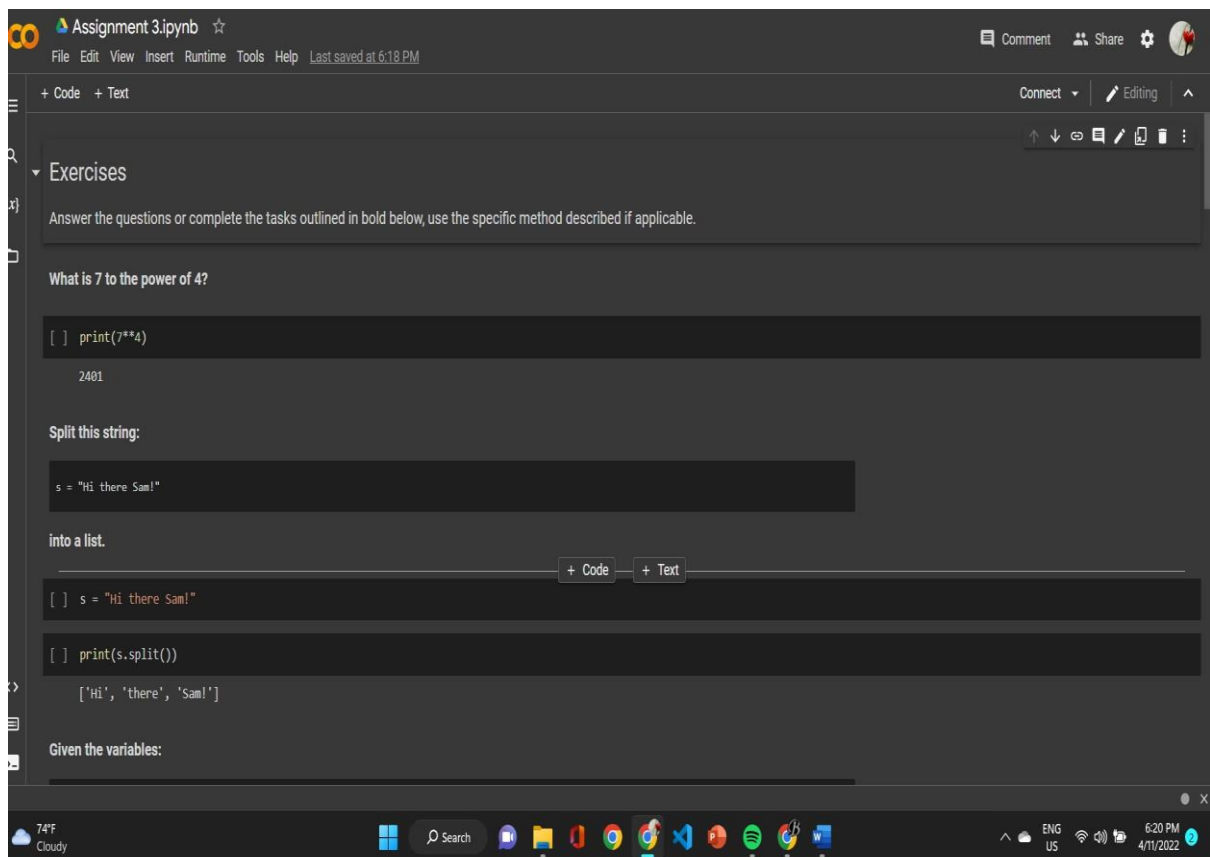
```
s.split()
```

Out[3]:

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

**Given the variables:**

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



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

The diameter of Earth is 12742 kilometers.

In [5]:

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

In [6]:

```
print("The diameter of {} is {} kilometers.".format(planet,diameter))
```

The diameter of Earth is 12742 kilometers.

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

In [7]:

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

In [14]:

```
lst[3][1][2][0]
```

Out[14]:

```
'hello'
```

```
Assignment 3.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 6:18 PM

+ Code + Text
Connect Editing ^

[ ] print(ss.split())

['Hi', 'there', 'Sam!']

Given the variables:

planet = "Earth"
diameter = 12742

Use .format() to print the following string:

The diameter of Earth is 12742 kilometers.

[ ] planet = "Earth"
diameter = 12742

[ ] print("The diameter of {} is {} kilometers.".format(planet, diameter))

The diameter of Earth is 12742 kilometers.

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

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

[ ] print(lst[3][1][2][0])
```

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

In [16]:

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

In [22]:

```
d['k1'][3]['tricky'][3]['target'][3]
```

Out[22]:

```
'hello'
```

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

In [23]:

```
# Tuple is immutable
```

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

[user@domain.com](mailto:user@domain.com)

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

In [24]:

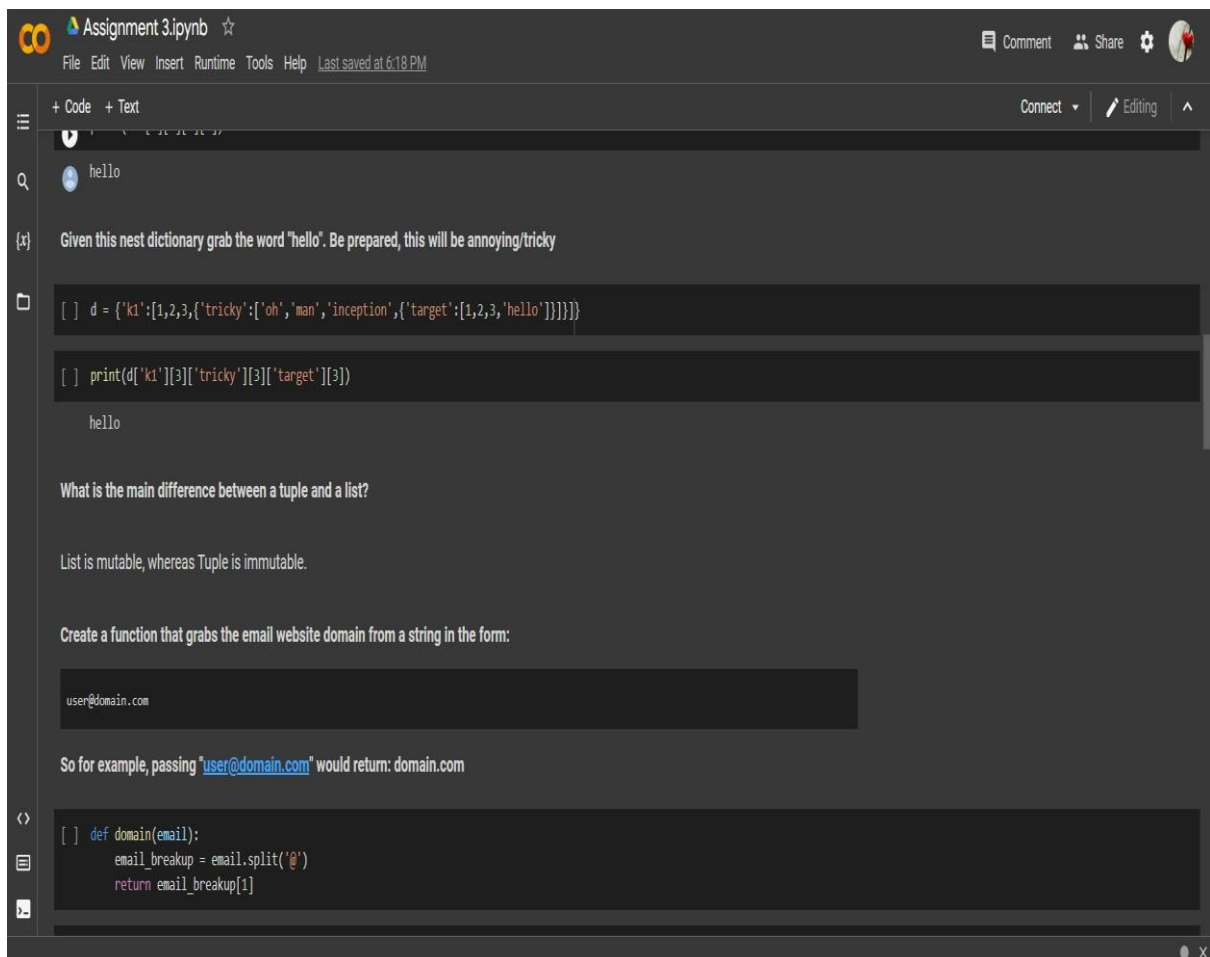
```
def domainGet(email):  
    return email.split('@')[-1]
```

In [26]:

```
domainGet('user@domain.com')
```

Out[26]:

```
'domain.com'
```



The screenshot shows a Jupyter Notebook window titled "Assignment 3.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for Comment, Share, and settings. The notebook content is displayed in a dark-themed editor. The first cell is a code cell containing a function definition and a test call. The second cell is a text cell containing a question and its answer. The third cell is a text cell containing a prompt and an example email address. The fourth cell is a code cell containing a function definition and its output.

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

```
[ ] print(d['k1'][3]['tricky'][3]['target'][3])
```

hello

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

List is mutable, whereas Tuple is immutable.

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](mailto:user@domain.com)" would return: domain.com

```
[ ] def domain(email):  
    email_breakup = email.split('@')  
    return email_breakup[1]
```

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.

In [27]:

```
def findDog(st):  
    return 'dog' in st.lower().split() In
```

[28]:

```
findDog('Is there a dog here?')
```

Out[28]:

True

The screenshot shows a Jupyter Notebook window titled "Assignment 3.ipynb". The interface includes a top bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with a "Last saved at 6:18 PM" timestamp. On the right, there are icons for "Comment", "Share", and a user profile. The notebook has two cells. The first cell is a code cell containing the function definition: 

```
def contains_dog(sentence):  
    words = sentence.split(' ')  
    k = 0  
  
    for i in range(len(words)):  
        words[i] = words[i].lower()  
  
    for i in words:  
        if i == 'dog':  
            k = 1  
            return True  
  
    if(k == 0):  
        return False
```

 Below the code is a text cell with the instruction: "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 second cell is a code cell containing the function call: 

```
[ ] print(contains_dog('There goes a dog'))
```

 Below the code is a text cell showing the output: "True". At the bottom, there is another text cell with the instruction: "Create a function that counts the number of times the word 'dog' occurs in a string. Again ignore edge cases."

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

In [30]:

```
def countDog(st):
```

```

count = 0    for word in
st.lower().split():    if word
== 'dog':
    count += 1
return count

```

In [31]: countDog('This dog runs faster than the other dog  
dude!')

Out[31]:

2

The screenshot shows a Jupyter Notebook titled "Assignment 3.ipynb". The interface includes a top bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with a "Last saved at 6:18 PM" timestamp. On the left, there are icons for a table of contents, search, and a file explorer. The main area contains a code cell with the following Python code:

```

def dog_count(sentence):
    words = sentence.split(' ')
    k = 0

    for i in range(len(words)):
        words[i] = words[i].lower()

    for i in words:
        if i == 'dog':
            k+=1

    return k

print(dog_count('There goes a dog and another dog'))

```

Below the code cell, the output of the execution is shown as the number "2". At the bottom of the notebook, there is a "Problem" section with a text description of a ticketing system based on speed and birthday status.

**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.

**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.**

In [4]:

```
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'
```

```
In [5]:
```

```
caught_speeding(81,True)
```

```
Out[5]: 'Small
```

```
Ticket'
```

```
In [6]:
```

```
caught_speeding(81,False)
```

```
Out[6]:
```

```
'Big Ticket'
```

The screenshot shows a Jupyter Notebook window titled "Assignment 3.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right, there are buttons for "Comment", "Share", and a user profile icon. Below the menu bar, the notebook is in "Code" mode, as indicated by the "+ Code" button. The left sidebar contains icons for a file explorer, search, and a "Problem" section. The main area displays a Python function named `caught_speeding` that takes `speed` and `is_birthday` as arguments. The function logic is as follows: if `is_birthday` is `True`, it subtracts 5 from the `speed`. Then, it checks the resulting `speeding` value: if greater than 80, it returns "Big Ticket"; if between 60 and 80 inclusive, it returns "Small Ticket"; otherwise, it returns "No Ticket". Below the function definition, a cell contains the code `print(caught_speeding(90, False))`, which has been executed, resulting in the output "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(90, False))
```

Big 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.

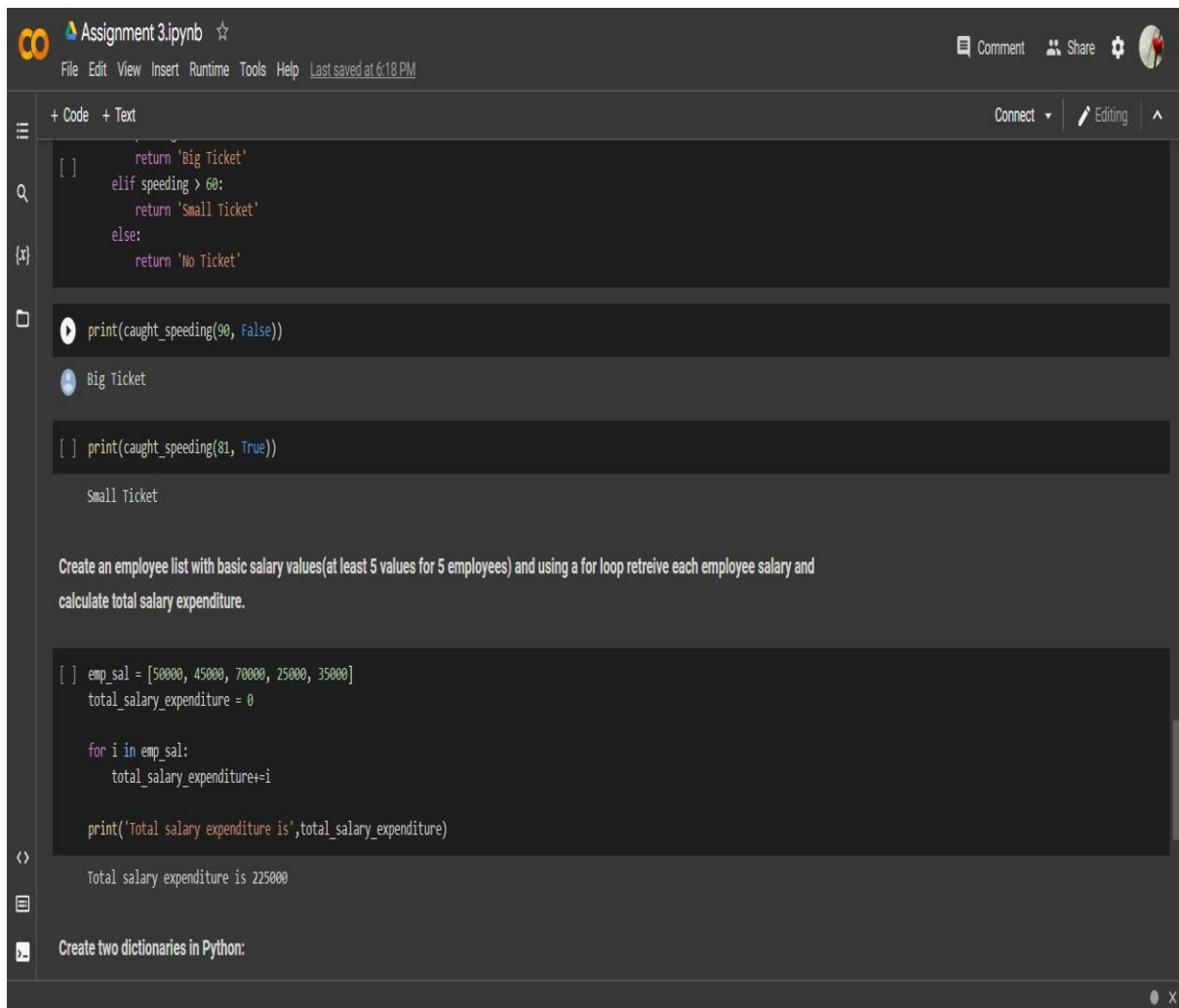


```
emp_sal = [50000, 45000, 70000, 25000, 35000] total_salary_expenditure  
= 0
```

```
for i in emp_sal:
```

```
    total_salary_expenditure+=i
```

```
print('Total salary expenditure is',total_salary_expenditure)
```



The screenshot shows a Jupyter Notebook titled "Assignment 3.ipynb". The interface includes a top bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with a "Last saved at 6:18 PM" timestamp. Below the menu bar, there are tabs for "+ Code" and "+ Text", and a "Connect" button. The main area displays a list of code cells. The first cell contains a function definition for `caught_speeding` that returns 'Big Ticket', 'Small Ticket', or 'No Ticket' based on the speed and whether the driver was caught. The second cell shows the function being called with `caught_speeding(90, False)`, resulting in the output "Big Ticket". The third cell shows the function being called with `caught_speeding(81, True)`, resulting in the output "Small Ticket". The fourth cell contains a text prompt: "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 fifth cell contains the Python code for creating the `emp_sal` list, initializing `total_salary_expenditure` to 0, and using a `for` loop to calculate the total salary expenditure. The output of this cell is "Total salary expenditure is 225000". The sixth cell contains a text prompt: "Create two dictionaries in Python:".

```
[ ]     return 'Big Ticket'  
    elif speeding > 60:  
        return 'Small Ticket'  
    else:  
        return 'No Ticket'  
  
[ ] print(caught_speeding(90, False))  
  
Big Ticket  
  
[ ] print(caught_speeding(81, True))  
  
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.  
  
[ ] emp_sal = [50000, 45000, 70000, 25000, 35000]  
    total_salary_expenditure = 0  
  
    for i in emp_sal:  
        total_salary_expenditure+=i  
  
    print('Total salary expenditure is',total_salary_expenditure)  
  
Total salary expenditure is 225000  
  
Create two dictionaries in Python:
```

**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.**

```
dict1 = {'Empid': 13, 'Empname': 'John Wick', 'Basicpay': 70000} dict2
```

```
= {'DeptName': 'Analytics', 'DeptId': 7}
```

```
dict3 = dict1 | dict2 print(dict3)
```

```
{'Empid': 13, 'Empname': 'John Wick', 'Basicpay': 70000, 'DeptName': 'Analytics', 'DeptId': 7}
```

Assignment 3.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 6:18 PM

Comment Share Settings

+ Code + Text Connect Editing ^

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.

```
[ ] emp_sal = [50000, 45000, 70000, 25000, 35000]
total_salary_expenditure = 0

for i in emp_sal:
    total_salary_expenditure+=i

print("Total salary expenditure is",total_salary_expenditure)
```

Total salary expenditure is 225000

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.

```
dict1 = {'Empid': 13, 'Empname': 'John Wick', 'Basicpay': 70000}
dict2 = {'DeptName': 'Analytics', 'DeptId': 7}

dict3 = dict1 | dict2
print(dict3)
```

```
{'Empid': 13, 'Empname': 'John Wick', 'Basicpay': 70000, 'DeptName': 'Analytics', 'DeptId': 7}
```