

Name: _____

Date: _____

Overview

	Lesson Plan
1	<ul style="list-style-type: none">- List & Dictionary- Functions- Random Library
2	<ul style="list-style-type: none">- Pandas- Bar Graph, Line Graph- Scatterplot, Pie Chart
3	<ul style="list-style-type: none">- Handling online CSV- Bar Graph, Line Graph- Multiplot
4	<ul style="list-style-type: none">- Data Analysis: % of o level cohort that progressed to post-secondary education

Python

Data Analytics Course – Dataset 1



Name: _____

Date: _____

Day 1

	Lesson Plan
1	<ul style="list-style-type: none">- List & Dictionary- Methods & functions- Random Library

Name: _____

Date: _____

Lesson 1.1

1.1 Learning – List & Dictionary

List

List Methods	Description
append()	Adds an element at the end of the list
remove()	Removes the first item with the specific value
pop()	Removes the element at the specific position
count()	Returns the number of elements with the specific value
sort()	Sorts the list

To learn more about the list methods, we will follow the guided tasks in writing. Please complete the output

Task 1: Create an empty list

<pre>mylist = [] print(mylist)</pre>	<u>Output</u>
--	---------------

Task 2: Create a list

<pre>mylist = ['apple', 'banana', 'cherry'] print(mylist)</pre>	<u>Output</u>
---	---------------

Name: _____

Date: _____

Lesson 1.1

Task 3: Update a list using append method

<pre>mylist = ['apple', 'banana', 'cherry'] mylist.append('durian') print(mylist)</pre>	<u>Output</u>
---	---------------

Task 4: update a list using extend method

<pre>mylist = ['apple', 'banana', 'cherry'] mylist.extend(['durian']) print(mylist)</pre>	<u>Output</u>
---	---------------

Task 5: Remove an element from a list by index

<pre>mylist = ['apple', 'banana', 'cherry'] mylist.pop(0) print(mylist)</pre>	<u>Output</u>
---	---------------

Task 6: Sorting a list

<pre>mylist = ["banana", "cherry", "apple"] mylist.sort() print(mylist)</pre>	<u>Output</u>
---	---------------

Name: _____

Date: _____

Lesson 1.1

Task 7: Sorting the list in reverse

<pre>mylist = ["banana", "cherry", "apple"] mylist.sort(reverse=True) print(mylist)</pre>	<u>Output</u>
---	---------------

Task 8: Getting length of list

<pre>mylist = ["banana", "cherry", "apple"] print(len(mylist)) print(mylist)</pre>	<u>Output</u>
--	---------------

Task 9: Looping through list by index

<pre>mylist = ["apple", "banana", "cherry"] for index in range(len(mylist)): print(mylist[index])</pre>	<u>Output</u>
---	---------------

Task 10: List indexing, slicing

<pre>height1 = 1.65 height2 = 1.81 height3 = 1.62 height4 = 1.76 height5 = 1.21 heightList = [height1, height2, height3, height4, height5] print(heightList) print(heightList[1]) print(heightList[-1]) print(heightList[1:3])</pre>	<u>Output</u>
--	---------------

Name: _____

Date: _____

Lesson 1.1

Task 11: Iterating through a list

<pre>list1 = [1,2,3,4,5,6,7] for numbers in list1: print(numbers)</pre>	<u>Output</u>
---	---------------

Task 12: Using range(stop)

<pre>list1 = [1,2,3,4,5,6,7] for numbers in range(len(list1)): print(list1[numbers])</pre>	<u>Output</u>
--	---------------

Task 13: Using range(start, stop)

<pre>list1 = [1,2,3,4,5,6,7] for numbers in range(1,3): print(list1[numbers])</pre>	<u>Output</u>
---	---------------

Task 14: Using range(start, stop, step)

<pre>list1 = [1,2,3,4,5,6,7] for numbers in range(1,len(list1),2): print(list1[numbers])</pre>	<u>Output</u>
--	---------------

Name: _____

Date: _____

Lesson 1.1**Dictionary**

Dictionary Methods	Description
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing dictionary's keys
values()	Returns a list containing dictionary's values
pop()	Removes the element with the specific key
copy()	Returns a copy of dictionary
update()	Updates the dictionary with the specific key-value pairs

To learn more about the dictionary methods, we will follow the guided tasks in writing. Please complete the output

Task 1: Create dictionary

<pre>student = { "name": "John Doe", "age": "21" } print(student)</pre>	<u>Output</u>
--	---------------

Task 2: Getting Items of dictionary

<pre>student = { "name": "John Doe", "age": "21" } print(list(student.items()))</pre>	<u>Output</u>
--	---------------

list() is used to typecase the student.items() into a list

Name: _____

Date: _____

Lesson 1.1

Task 3: Getting Keys of dictionary

<pre>student = { "name": "John Doe", "age": "21" } print(list(student.keys()))</pre>	<u>Output</u>
---	---------------

Task 4: Getting Values of dictionary

<pre>student = { "name": "John Doe", "age": "21" } print(list(student.values()))</pre>	<u>Output</u>
---	---------------

Task 5: Update dictionary

<pre>student = { "name": "John Doe", "age": "21" } student.update({"result": "pass"}) print(student)</pre>	<u>Output</u>
---	---------------

Name: _____

Date: _____

Lesson 1.1

Task 6: Remove from dictionary

<u>Output</u>
<pre>student = { "name": "John Doe", "age": "21", "result": "unknown" } student.pop("result") # Pop by key print(student)</pre>

Task 7: Modify value from dictionary

<u>Output</u>
<pre>student = { "name": "John Doe", "age": "21", "result": "pass" } student["result"] = "fail" # By key print(student)</pre>

Other ways of modifying the value in dictionary is `student.update({"result": "fail"})`

Task 8: Modify value in dictionary

<u>Output</u>
<pre>student = { "name": "John Doe", "age": "21", "result": "pass" } student.update({"result": "fail"}) # By key print(student)</pre>

Name: _____

Date: _____

Lesson 1.1**1.1 Practice – List & Dictionary****Exercise 1**

The Fibonacci sequence is the series of numbers where each number is the sum of the two preceding numbers.

0	1
---	---

0	1	1
---	---	---

0	1	1	2
---	---	---	---

0	1	1	2	3
---	---	---	---	---

Write a python code that print the first 10 terms of Fibonacci sequence in a list.

Input
10
Output
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

Exercise 2

Write a Python program to split a given dictionary of lists into list of dictionaries.

Input
{'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}
Output
[{'Science': 88, 'Language': 77}, {'Science': 89, 'Language': 78}, {'Science': 62, 'Language': 84}, {'Science': 95, 'Language': 80}]

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 1.1

Exercise 1

The Fibonacci sequence is the series of numbers where each number is the sum of the two preceding numbers.

0	1
---	---

0	1	1
---	---	---

0	1	1	2
---	---	---	---

0	1	1	2	3
---	---	---	---	---

Write a python code that print the first 10 terms of Fibonacci sequence in a list.

Input
10
Output
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

Step 1: Understanding the problem

- Starting point is [0,1]
- To get the next term, we need to add 2 of the previous

Step 2: Creating the starting point

```
fib = [0,1]
```

Step 3: Looping through the string

```
for i in range(10-2):  
    fib.append(fib[-1] + fib[-2])
```

Step 4: Display

```
print(fib)
```

Name: _____

Date: _____

Lesson 1.1

Exercise 2

Write a Python program to split a given dictionary of lists into list of dictionaries.

Input
{'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}
Output
[{'Science': 88, 'Language': 77}, {'Science': 89, 'Language': 78}, {'Science': 62, 'Language': 84}, {'Science': 95, 'Language': 80}]

Step 1: Understanding the problem

- We need to loop through the list within 'Science' and 'Language'
- We need to reconstruct in the format of {'Science': value , 'Language': value}

Step 2: Creating the input

```
data = {'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}
```

Step 3: Creating the empty list

```
l = [] # Empty list
```

Step 4: Looping through the string

```
for d in range(len(data['Science'])):  
    l.append({'Science': data['Science'][d], 'Language': data['Language'][d]})
```

Step 5: Displaying the list

```
print(l)
```

Name: _____

Date: _____

Lesson 1.2

1.2 Learning – Recap on function

What is a function?

- A function is a block of organised, reusable code that is used to perform a single, related action
- Parameters are used to pass data into a function
- Functions are classified as either built-in functions or user-defined functions

Examples of common built-in functions

Common Built-in Function	Description
max()	Adds an element at the end of the list
min()	Removes the first item with the specific value
sum()	Removes the element at the specific position
len()	Returns the number of elements with the specific value

Rules for creating user-defined function

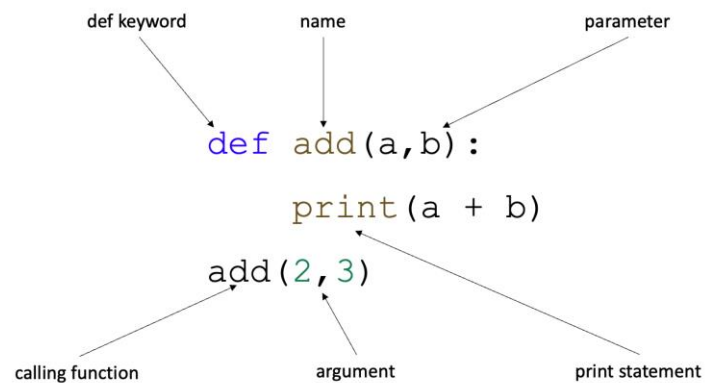
- Function blocks begin with the keyword ``def`` followed by the function name and parentheses `()`. The keyword ``def`` marks the start of the function header.
- Function name is used to uniquely identify it, should not be used in conjunction with variable names
- The code block within every function starts with a colon `(:)` and is indented
- From the calling side, data is passed to the function as argument
- From the function side, data is passed to the function as parameter

Name: _____

Date: _____

Lesson 1.2

Anatomy of a function



Example: Understanding the anatomy of a function

```
def add(a,b):
    print(a + b)
add(2,3)
```

Question	Answer
What is the function name?	<i>add</i>
What is the function argument?	2,3
What is the function parameters?	a,b
What is the output?	5
Are we calling the function or defining it when we use def ?	defining

Name: _____

Date: _____

Lesson 1.2

Task 1: Understanding the anatomy of a function

```
def add(a,b,c):  
    print(a + b + c)  
add(1,2,3)
```

Question	Answer
<i>What is the function name?</i>	
<i>What is the function argument?</i>	
<i>What is the function parameters?</i>	
<i>What is the output?</i>	
<i>Are we calling the function or defining it when we use def?</i>	

Task 2: Understanding the anatomy of a function

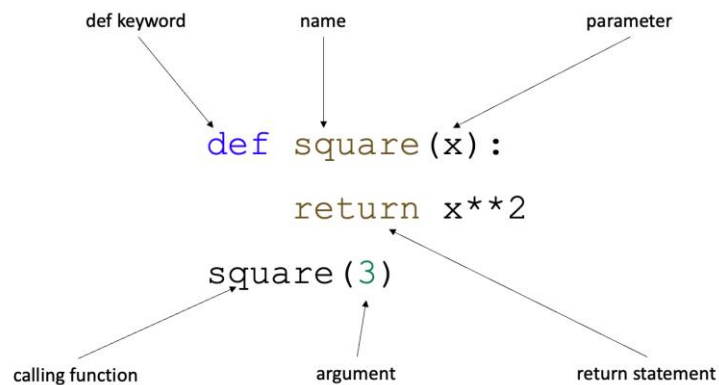
```
def pow(a,b):  
    print(a ** b)  
pow(2,5)
```

Question	Answer
<i>What is the function name?</i>	
<i>What is the function argument?</i>	
<i>What is the function parameters?</i>	
<i>What is the output?</i>	
<i>Are we calling the function or defining it when we use def?</i>	

Name: _____

Date: _____

Lesson 1.2



Example: Understanding the anatomy of a function with return

```

def square(x):
    return x**2

square(3)
    
```

Question	Answer
What is the function name?	square
What is the function argument?	3
What is the function parameters?	x
What is the output?	-
Are we calling the function or defining it when we use def ?	defining

Name: _____

Date: _____

Lesson 1.2

Complete the following tasks to get the output. Do look at the remarks.

Task 1: Returning only within the function

<u>Code</u>	<u>Output</u>	<u>Remarks</u>
<pre>def add(x, y): return x + y add(3, 5)</pre>		<p>There is no print statement.</p> <p>add(3, 5) will contain a value</p>

Task 2: Printing only within the function

<u>Code</u>	<u>Output</u>	<u>Remarks</u>
<pre>def add(x, y): print(x + y) add(3, 5)</pre>		<p>There is only 1 print statement.</p> <p>add(3, 5) will contain None</p>

Task 3: Returning with a printing outside the function

<u>Code</u>	<u>Output</u>	<u>Remarks</u>
<pre>def add(x, y): return x + y print(add(3, 5))</pre>		<p>There is only 1 print statement.</p> <p>add(3, 5) will contain a value</p>

Task 4: Printing with a printing outside the function

<u>Code</u>	<u>Output</u>	<u>Remarks</u>
<pre>def add(x, y): print(x + y) print(add(3, 5))</pre>		<p>A function will always return None unless specified</p>

Name: _____

Date: _____

Lesson 1.2

1.2 Practice – Recap on function

Exercise 1

Write a Python code to emulate a basic calculator in a command line interface (CLI).

The user requirements are as follows:

1. As a user, I want to add numbers upon request - x, y
2. As a user, I want to multiply numbers upon request - x, y

Exercise 2

Additional user requirement is as follows:

1. As a user, I want to determine if a number is even or odd

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 1.2

Exercise 1

Write a Python code to emulate a basic calculator in a command line interface (CLI).

The user requirements are as follows:

1. As a user, I want to add numbers upon request - x, y
2. As a user, I want to multiply numbers upon request - x, y

Step 1: Understanding the problem

- Create specific functions to add & multiply numbers
- To add or multiply, we need to request for 2 numbers

Step 2: Create the add function

```
def add():  
    x = int(input("Enter x:\t"))  
    y = int(input("Enter y:\t"))  
    print(f"{x} + {y} = {x+y}")
```

Step 2: Create the multiply function

```
def mul():  
    x = int(input("Enter x:\t"))  
    y = int(input("Enter y:\t"))  
    print(f"{x} * {y} = {x*y}")
```

Step 3: Create the loop with termination case

```
while True:  
    c = input() # input()  
    if c == 'quit':  
        break
```

While True is used so that the user can make infinite number of requests. However, when the user wants to end, he/she needs to only enter 'quit'.

Name: _____

Date: _____

Lesson 1.2

Step 4: Adding cases

```
elif c == 'add':  
    add()  
  
elif c == 'mul':  
    mul()
```

IF ELSE ELIF are used to denote the various cases

Step 5: Rejecting invalid

```
else:  
    print("invalid")
```

What if the user wrote a command that is not one of the cases? Then by default, we should reject that request and inform the user

Name: _____

Date: _____

Lesson 1.2

Exercise 2

Additional user requirement is as follows:

1. As a user, I want to determine if a number is even or odd

Step 1: Create function – isEven

```
def isEven():  
    x = int(input("Enter number:\t"))  
    if x % 2 == 0:  
        return True  
    else:  
        return False
```

*For every **even** number, when divided by 2 will have a remainder of 0*

Step 2: Create function – isOdd

```
def isOdd():  
    x = int(input("Enter number:\t"))  
    if x % 2 == 1:  
        return True  
    else:  
        return False
```

*For every **odd** number, when divided by 2 will have a remainder of 1*

Step 3: Adding new cases

```
elif c == 'odd':  
    isOdd()  
  
elif c == 'even':  
    isEven()
```

Name: _____

Date: _____

Lesson 1.3

1.3 Learning – Recap on Random Library

What is a module?

Consider a module to be the same as a code library. A file containing a set of functions you want to include in your application. Any text file with the .py extension containing Python Code is basically a module.

What is a library?

To begin, let us first talk about libraries and what they are. A library is a collection of modules that contains function for the use by other programs. They may contain some data values in them, and are often used to make a program's life easier

Python has built-in modules - Primarily, OS module, Sys module, Math module, Statistics module, Collections module & Random Module.

Objective

In this section, we will focus on the random library. The random module is a built-in module to generate the pseudo-random variables. It can also be used to perform some action randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly.

Python

Data Analytics Course – Dataset 1



Name: _____

Date: _____

Lesson 1.3

Task 1: Learning about random library

```
# To find out more  
help(random.random)
```

Fill up the description after using help() to look up each methods

Method	Description
random.randint	
random.randrange	
random.choice	
random.shuffle	

Python

Data Analytics Course – Dataset 1



Name: _____

Date: _____

Lesson 1.3

In the next few tasks, use the methods you've learn to write Python codes

Task 2:

Write a Python Code to generate a random integer between 1 to 6.

Insert Code here

Task 3:

Write a Python Code to generate a random float between 0 to 1.

Insert Code here

Task 4:

Write a Python Code to generate a random alphabet.

```
import string
lst = string.ascii_letters
print(lst)
```

Insert Code here

lst will contains the alphabets, how do we then pick a random alphabet?

Name: _____

Date: _____

Lesson 1.3

1.3 Practice – Recap on Random library

Exercise 1

Write a Python code to simulate the behaviour of a random fair 6-sided dice.

The user requirements are as follows:

1. As a user, I want a function, `roll_dice()`, that returns a **random** number between 1 to 6
2. As a user, I want a function, `roll_double()`, that returns the sum of 2 `roll_dice()`
3. As a user, I want to loop the function 100 times & store the results in a list

Exercise 2

Write a function, `display_stats()` that takes in the list with size 100 to print the following

1. **Max** value of the list
2. **Min** value of the list
3. **Mean** value of the list
4. **Median** value of the list

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 1.3

Exercise 1

Write a Python code to simulate the behaviour of a random fair 6-sided dice.

The user requirements are as follows:

1. As a user, I want a function, *roll_dice()*, that returns a **random** number between 1 to 6
2. As a user, I want a function, *roll_double()*, that returns the sum of 2 *roll_dice()*
3. As a user, I want to loop the function 100 times & store the results in a list

Step 1: Understanding the problem

- Create specific functions to add & multiply numbers
- To add or multiply, we need to request for 2 numbers

Step 2: Import relevant function

```
import random
```

Step 3: Creating roll_dice()

```
def roll_dice():  
    return random.randint(1,6)
```

The function will use the random.randint(1,6) to generate a random number between 1 and 6

Step 4: Creating roll_double()

```
def roll_double():  
    return roll_dice() + roll_dice()
```

The function will call roll_dice() twice

Step 5: Creating the loop

```
l = []  
for i in range(100):  
    l.append(roll_double())
```

Run loop for 100 times and store results

Name: _____

Date: _____

Lesson 1.2

Exercise 2

Write a function, `display_stats()` that takes in the list with size 100 to print the following

1. **Max** value of the list
2. **Min** value of the list
3. **Mean** value of the list
4. **Median** value of the list

Step 4: Creating the display of the stats

```
def display_stats(lst):  
    print('Max =', max(lst))  
    print('Min =', min(lst))  
    print('Mean =', sum(lst)/len(lst))  
    lst.sort()  
    print('Median =', (lst[len(lst)//2] + lst[len(lst)//2 - 1]) / 2)  
  
display_stats(l)
```

To get the mean, we need to take the sum of total divided by the number of elements in list

To get the median, we need to get the middle of a sorted list

Python

Data Analytics Course – Dataset 1



Name: _____

Date: _____

Day 2

	Lesson Plan
2	<ul style="list-style-type: none">- Pandas- Bar Graph, Line Graph- Scatterplot, Pie Graph

Name: _____

Date: _____

Lesson 2.1

2.1 Learning – Revisiting Pandas

When it comes to creating DataFrame with Pandas in Python, we can always create using a list of dictionaries or a dictionary containing lists.

Code	Output
<pre>import pandas as pd d = [{"name": "Alice", "age": 10}, {"name": "Bobby", "age": 11}, {"name": "Carl", "age": 14}] df = pd.DataFrame(d) df</pre>	

<pre> name age 0 Alice 10 1 Bobby 11 2 Carl 14 </pre>	<p>Looking at the headers</p> <p><i>df.columns</i></p>
<pre> name age 0 Alice 10 1 Bobby 11 2 Carl 14 </pre>	<p>Looking at the indexes</p> <p><i>df.index</i></p>
<pre> name age 0 Alice 10 1 Bobby 11 2 Carl 14 </pre>	<p>Looking at the first row, index = 0</p> <p><i>df.iloc[0]</i></p>
<pre> name age 0 Alice 10 1 Bobby 11 2 Carl 14 </pre>	<p>Looking at the column 'name'</p> <p><i>df['name']</i></p>

Name: _____

Date: _____

Lesson 2.1

Task 1: List of Dictionary Entries

<u>Code</u>	<u>Output</u>
<pre>import pandas as pd d = [{"name": "Alice", "age": 10}, {"name": "Bobby", "age": 11}, {"name": "Carl", "age": 14},] df = pd.DataFrame(d) df</pre>	

Task 2: Understanding a DataFrame

<u>Code</u>	<u>Output</u>
<pre>d = [{"name": "Alice", "age": 10}, {"name": "Bobby", "age": 11}, {"name": "Carl", "age": 14},] df = pd.DataFrame(d) print(f""" headers:\n{df.columns}\n index:\n{df.index}\n first row:\n{df.iloc[0]}\n columns `Name`: \n{df['name']} \n """)</pre>	

Name: _____

Date: _____

Lesson 2.1

Task 3: Dictionary containing lists

<u>Code</u>	<u>Output</u>
<pre>import pandas as pd d = { "name": ["Apple", "Pear", "Cucumber"], "quantity": [3, 2, 5] } df = pd.DataFrame(d) df</pre>	

Task 4: Grouping Data by Specific Column

<u>Code</u>	<u>Output</u>
<pre># Grouping Data d = [{"name": "Apple", "quantity": 3, 'type': 'Food'}, {"name": "Pear", "quantity": 2, 'type': 'Food'}, {"name": "Grape", "quantity": 5, 'type': 'Food'}, {"name": "Kettle", "quantity": 1, 'type': 'Appliance'}, {"name": "Pan", "quantity": 2, 'type': 'Appliance'}, {"name": "Fan", "quantity": 3, 'type': 'Appliance'},] df = pd.DataFrame(d) df.groupby(by=['type']).sum()</pre>	

When you `df.groupby(by=['type']).sum()`, type will become the index of the DataFrame.

Name: _____

Date: _____

Lesson 2.1

Task 5: Resetting the index

<u>Code</u>	<u>Output</u>
<pre>d = [{"name": "Apple", "quantity": 3, 'type': 'Food'}, {"name": "Pear", "quantity": 2, 'type': 'Food'}, {"name": "Cucumber", "quantity": 5, 'type': 'Food'}, {"name": "Kettle", "quantity": 1, 'type': 'Appliance'}, {"name": "Pan", "quantity": 2, 'type': 'Appliance'}, {"name": "Fan", "quantity": 3, 'type': 'Appliance'},] df = pd.DataFrame(d) df = df.groupby(by=['type']).sum() df.reset_index()</pre>	

When you `df.groupby(by=['type']).sum()`, 'type' will become the index of the DataFrame. However, you do not want to have 'type' as the index. You can reset the index back to 0-indexing by using `df.reset_index()`

Name: _____

Date: _____

Lesson 2.1

Task 6: Selecting row

<u>Code</u>	<u>Output</u>
<pre>d = [{"name": "Apple", "quantity": 3, 'type': 'Food'}, {"name": "Pear", "quantity": 2, 'type': 'Food'}, {"name": "Grape", "quantity": 5, 'type': 'Food'}, {"name": "Kettle", "quantity": 1, 'type': 'Appliance'}, {"name": "Pan", "quantity": 2, 'type': 'Appliance'}, {"name": "Fan", "quantity": 3, 'type': 'Appliance'},] df = pd.DataFrame(d) df = df[df['name'] == 'Grape'] df</pre>	

df[df['name'] == 'Grape'] filter out all the data that has 'name' as Grape

Name: _____

Date: _____

Lesson 2.1**Task 7: Updating the row**

<u>Code</u>	<u>Output</u>
<pre>d = [{"name": "Apple", "quantity": 3, 'type': 'Food'}, {"name": "Pear", "quantity": 2, 'type': 'Food'}, {"name": "Grape", "quantity": 5, 'type': 'Food'}, {"name": "Kettle", "quantity": 1, 'type': 'Appliance'}, {"name": "Pan", "quantity": 2, 'type': 'Appliance'}, {"name": "Fan", "quantity": 3, 'type': 'Appliance'},] df = pd.DataFrame(d) df.loc[df['type'] == 'Appliance', ['quantity']] = 0 df</pre>	

Task 8: Inserting new the column

<u>Code</u>	<u>Output</u>
<pre>d = [{"name": "Apple", "quantity": 3}, {"name": "Pear", "quantity": 2}, {"name": "Cucumber", "quantity": 5},] df = pd.DataFrame(d) df['profit'] = df['quantity'] df['profit'] *= 1.2 df</pre>	

Name: _____

Date: _____

Lesson 2.1**2.1 Practice – Pandas****Exercise**

Using the above methods, create 2 DataFrame for Pokemon Go Users & Roblox users respectively.

Pokémon Go users

Year	Users
2016	232 million
2017	65 million
2018	147 million
2019	153 million
2020	166 million

Roblox users

Year	Users
2017	35 million
2018	80 million
2019	100 million
2020	150 million

References

<https://www.businessofapps.com/data/pokemon-go-statistics/>

<https://www.businessofapps.com/data/roblox-statistics/>

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 2.1

Pokémon Go users

Year	Users
2016	232 million
2017	65 million
2018	147 million
2019	153 million
2020	166 million

Exercise – pokemonGoUsers

```
import pandas as pd
pokemonGoUsers = {
    'year': [2016,2017,2018,2019,2020],
    'users': [232, 65, 147, 153, 166]
}
df = pd.DataFrame(pokemonGoUsers)
print("PokemonGoUsers")
df
```

In this example, we created a dictionary containing list for each column

Name: _____

Date: _____

Lesson 2.1**Roblox users**

Year	Users
2017	35 million
2018	80 million
2019	100 million
2020	150 million

Exercise - robloxUsers

```
import pandas as pd
robloxUsers = {
    'year': [2017,2018,2019,2020],
    'users': [35,80, 100, 150]
}
df = pd.DataFrame(robloxUsers)
print("RobloxUsers")
df
```

In this example, we created a dictionary containing list for each column

Name: _____

Date: _____

Lesson 2.2**2.2 Learning – Bar Graph, Line Graph**

The table below shows the number of fruits sold across 7 days.

Item	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Apple	20	21	16	15	18	12	18
Orange	10	13	9	20	2	23	2
Pear	15	11	19	21	22	26	12
Banana	12	23	14	6	13	16	27

Write a Python Code using Pandas Library to do the following

1. Load the data into a DataFrame
2. Select the row containing the **Pear**
3. Plot a bar graph for **Pear**
4. Select the row containing **Apple**
5. Plot a line graph for **Apple**

Task 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Task 2: Load in Dataset into a DataFrame

```
items = {
    'apple': [20, 21, 16, 15, 18, 12, 18],
    'orange': [10, 13, 9, 20, 2, 23, 2],
    'pear': [15, 11, 19, 21, 22, 26, 12],
    'banana': [12, 23, 14, 6, 12, 16, 27]
}
df = pd.DataFrame(items)
```

Name: _____

Date: _____

Lesson 2.2

Item	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Pear	15	11	19	21	22	26	12

1. Select the row containing the **Pear**
2. Plot a bar graph for **Pear**

Task 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1, figsize=(10,5))
```

Task 4: Setting the width of bar & indexes along the x-axis

```
w = 0.2  
x = np.asarray([i+1 for i in range(7)])
```

*[i+1 for i in range(7)] is equivalent to [1, 2, 3, 4, 5, 6, 7]***Task 5:** Plot bar graph

```
ax1.bar(x, df['pear'], width = w)
```

Task 6: Set the title, x-label & y-label

```
ax1.set_title("Pear")  
ax1.set_xlabel("Day")  
ax1.set_ylabel("Sales")
```

Name: _____

Date: _____

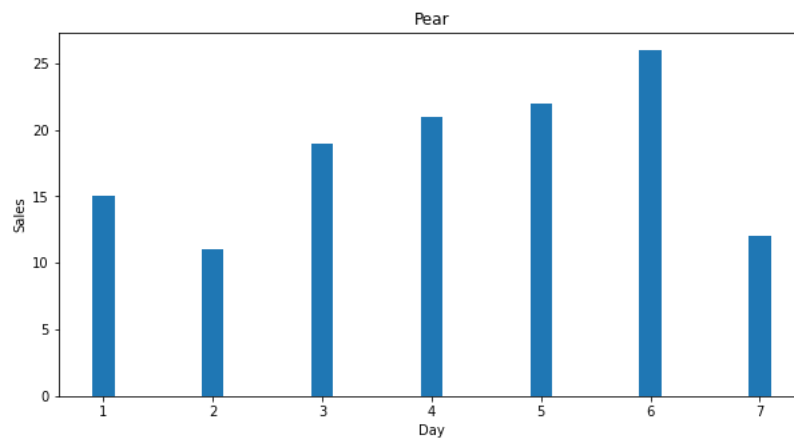
Lesson 2.2

Task 7: Set x-ticks

```
ax1.set_xticks(x)
```

Task 8: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 2.2

Item	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Apple	20	21	16	15	18	12	18

3. Select the row containing **Apple**
4. Plot a line graph for **Apple**

Task 3: Create the figure and axes

```
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 5))
```

Task 4: Setting the indexes along the x-axis

```
x = np.asarray([i+1 for i in range(7)])
```

[i+1 for i in range(7)] is equivalent to [1, 2, 3, 4, 5, 6, 7]

Task 5: Plot the line graph

```
ax1.plot(x, df['apple'])
```

Task 6: Set the title, x-label & y-label

```
ax1.set_title("Apple")  
ax1.set_xlabel("Day")  
ax1.set_ylabel("Sales")
```

Name: _____

Date: _____

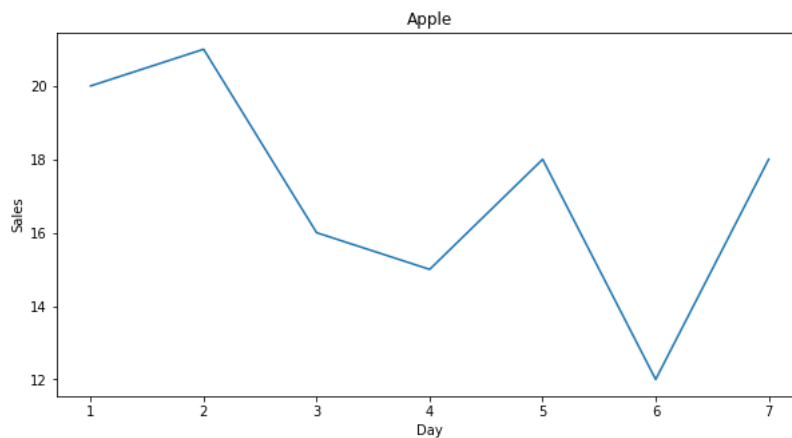
Lesson 2.2

Task 7: Set x-ticks

```
ax1.set_xticks(x)
```

Task 8: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 2.2**2.2 Practice – Bar Graph, Line Graph**

Write a Python Code to plot the Pokémon Go users and Roblox users as a bar graph and line graph respectively

Pokémon Go users

Year	Users
2016	232 million
2017	65 million
2018	147 million
2019	153 million
2020	166 million

Roblox users

Year	Users
2017	35 million
2018	80 million
2019	100 million
2020	150 million

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 2.2

Pokémon Go users

Year	Users
2016	232 million
2017	65 million
2018	147 million
2019	153 million
2020	166 million

Task 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Task 2: Load in Dataset into a DataFrame

```
pokemonGoUsers = {
    'year': [2016, 2017, 2018, 2019, 2020],
    'users': [232, 65, 147, 153, 166]
}
df = pd.DataFrame(pokemonGoUsers)
```

Task 3: Create the figure and axes

```
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 5))
```

Name: _____

Date: _____

Lesson 2.2

Task 4: Setting the width of bar & indexes of the bar on the x-axis

```
w = 0.2  
x = np.asarray(df['year'])
```

Task 5: Plot the bar graph

```
ax1.bar(x, df['users'], width = w)
```

Task 6: Set the title, x-label & y-label

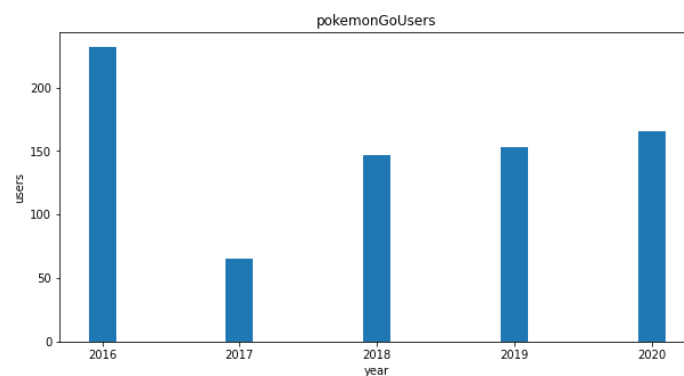
```
ax1.set_title("pokemonGoUsers")  
ax1.set_xlabel("year")  
ax1.set_ylabel("users")
```

Task 7: Set x-ticks

```
ax1.set_xticks(df['year'])
```

Task 8: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 2.2**Roblox users**

Year	Users
2017	35 million
2018	80 million
2019	100 million
2020	150 million

Task 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Task 2: Load in Dataset into a DataFrame

```
robloxUsers = {
    'year': [2017,2018,2019,2020],
    'users': [35,80, 100, 150]
}
df = pd.DataFrame(robloxUsers)
```

Task 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1, figsize=(10,5))
```

Name: _____

Date: _____

Lesson 2.2

Task 4: Plot the line graph

```
ax1.plot(df['year'], df['users'])
```

Task 5: Set the title, x-label & y-label

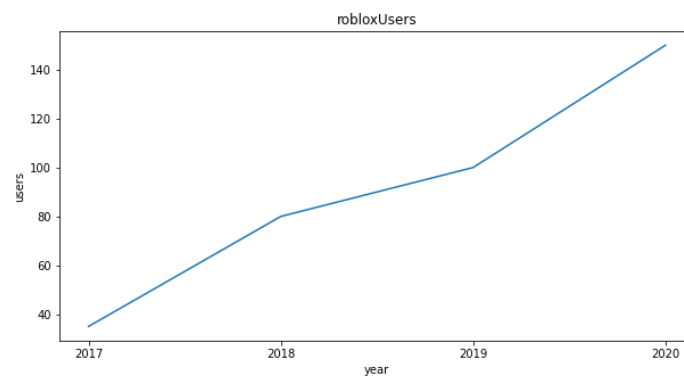
```
ax1.set_title("robloxUsers")  
ax1.set_xlabel("year")  
ax1.set_ylabel("users")
```

Task 6: Set x-ticks

```
ax1.set_xticks(df['year'])
```

Task 7: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 2.3

2.3 Learning – Scatterplot, Pie Graph

Let's continue with writing **scatterplot** & pie graph

X	1	2	3	4	5	6	7	8	9	10
Y	103	101	99	100	100	95	95	96	93	90

Task 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
```

Task 2: Create the data

```
x = list(range(10))
y = [103, 101, 99, 100, 100, 95, 95, 96, 93, 90]

df = pd.DataFrame({'x':x, 'y':y})
```

Task 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1, figsize=(10,5))
```


Name: _____

Date: _____

Lesson 2.3

Task 4: Plot the scatterplot graph

```
ax1.scatter(df['x'], df['y'])
```

Task 5: Set the title, x-label & y-label

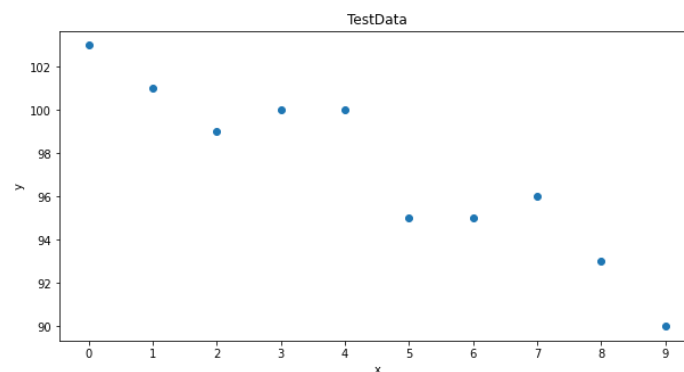
```
ax1.set_title("TestData")  
ax1.set_xlabel("x")  
ax1.set_ylabel("y")
```

Task 6: Set x-ticks

```
ax1.set_xticks(df['x'])
```

Task 7: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 2.3

Let's continue with writing scatterplot & pie graph

Names	Value	Colour
Apple	103	r
Orange	101	g
Pear	55	c
Durian	10	b
Mango	82	y

Task 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
```

Task 2: Create the data

```
z = [103, 101, 55, 10, 82]
df = pd.DataFrame({'z': z})
names = ["apple", "orange", "pear", "durian", "mango"]
mycolors = ['r', 'g', 'c', 'b', 'y']
```

Task 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1, figsize=(10,5))
```

Name: _____

Date: _____

Lesson 2.3

Task 4: Plot the pie chart

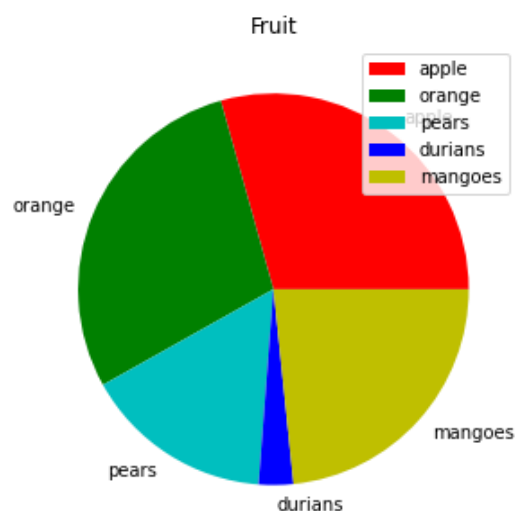
```
ax1.pie(z, labels=names, colors=mycolors)
```

Task 5: Set the title, x-label & y-label

```
ax1.set_title("Fruit")
```

Task 6: Display the plot

```
plt.show()
```



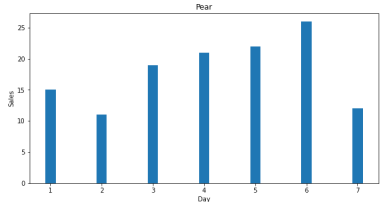
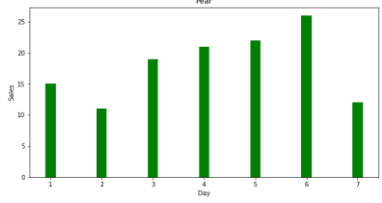
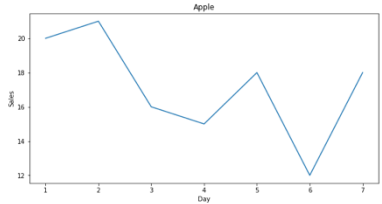
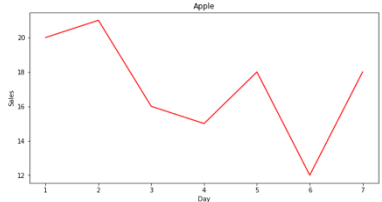
Name: _____

Date: _____

Lesson 2.3

Decorating the graphs – uses examples from the previous section

Setting Colour

<pre>ax1.bar(x, df['Pear'], width = w)</pre>	
<pre>ax1.bar(x, df['Pear'], width = w, color="Green")</pre>	
<pre>ax1.plot(x, df['Apple'])</pre>	
<pre>ax1.plot(x, df['Apple'], color="Red")</pre>	

Name: _____

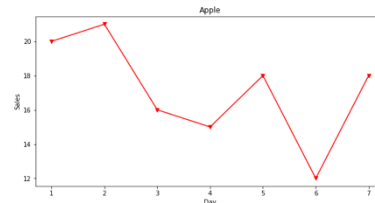
Date: _____

Lesson 2.3

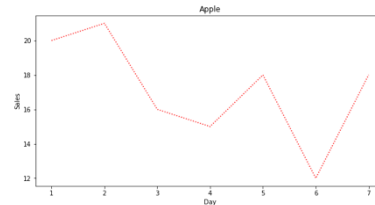
Decorating the graphs – uses examples from the previous section

Setting Markers & Linestyle (Only Applicable to line graph not bar graph)

```
ax1.plot(x, df['Apple'], color="Red",
marker='v')
```



```
ax1.plot(x, df['Apple'], color="Red",
linestyle=':')
```



Name: _____

Date: _____

Lesson 2.3

Decorating the graphs – List

Marker	Description
'o'	Circle
'*'	Star
'.'	Point
','	Pixel
'x'	X
'X'	X (filled)
'+'	Plus
'p'	Plus (filled)
's'	Square
'D'	Diamond
'd'	Diamond (thin)
'p'	Pentagon
'H'	Hexagon
'h'	Hexagon
'v'	Triangle Down
'^'	Triangle Up
'<'	Triangle Left
'>'	Triangle Right
'1'	Tri Down

Name: _____

Date: _____

Lesson 2.3

Decorating the graphs – List

Style	Or
'solid' (default)	'_'
'dotted'	':'
'dashed'	'--'
'dashdot'	'-.'
'None'	'' or ''

Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

Name: _____

Date: _____

Day 3

	Lesson Plan
3	<ul style="list-style-type: none">- Handling online CSV- Bar graph, Line graph- Multiplot

Name: _____

Date: _____

Lesson 3.1

3.1 Learning – Handling Online Data with Pandas

Task 1: Import online dataset

```
import pandas as pd

url = "https://raw.githubusercontent.com/tlc-
datascience/datasets/main/Spotify_2010_2019.csv"

df = pd.read_csv(url)
df
```

When you enter the URL into the browser, you will find a CSV file containing the raw contents. Read CSV get the data. Alternatively, you can replace the URL to the path to your CSV file. In Google Colab, it will usually be the name of the file.

Task 2: Understanding the DataFrame

```
df.info()
```

Use info to get the necessary information

Task 3: Filter out the Columns

```
df = df[['title', 'artist', 'top genre', 'year']]
df
```

You can filter out the columns in the DataFrame.

Name: _____

Date: _____

Lesson 3.1

From the given dataset, we want to find out the following

1. How many songs does **Ed Sheeran** have in the dataset?
2. What is the top genre of the songs by **Ed Sheeran**?
3. What is the first song by the **Ed Sheeran**?
4. Which year is the song **Shape of You** by the **Ed Sheeran** produced?

Task 4: Getting **Ed Sheeran** data

```
df[df['artist'] == 'Ed Sheeran']
```

Task 5: How many songs does **Ed Sheeran** have in the dataset

```
len(df[df['artist'] == 'Ed Sheeran'])
```

Task 6: What is the top genre of the song by **Ed Sheeran**

```
df[df['artist'] == 'Ed Sheeran'][['top genre']].iloc[0]
```

Task 7: What is the first song by **Ed Sheeran**

```
df[df['artist'] == 'Ed Sheeran'][['title']].iloc[0]
```

Task 8: Which year is the song **Shape of You** by the **Ed Sheeran** produced

```
df[df['title'] == 'Shape of You']['year']
```

Name: _____

Date: _____

Lesson 3.1

3.1 Practice – Handling Online Data with Pandas

From the given dataset, we want to find out the following

1. How many songs does **The Chainsmokers** have in the dataset?
2. What is the top genre of the songs by **The Chainsmokers**?
3. What is the first song by the **The Chainsmokers**?
4. Which year is the song **Closer** by the **The Chainsmokers** produced?

Guided solution found in the next few pages

Name: _____

Date: _____

Lesson 3.1

From the given dataset, we want to find out the following

1. How many songs does **The Chainsmokers** have in the dataset?
2. What is the top genre of the songs by **The Chainsmokers**?
3. What is the first song by the **The Chainsmokers**?
4. Which year is the song **Closer** by the **The Chainsmokers** produced?

Task 4: Getting The Chainsmokers data

```
df[df['artist'] == 'The Chainsmokers']
```

Task 5: How many songs does The Chainsmokers have in the dataset

```
len(df[df['artist'] == 'The Chainsmokers'])
```

Task 6: What is the top genre of the song by The Chainsmokers

```
df[df['artist'] == 'The Chainsmokers']['top genre'].iloc[0]
```

Task 7: What is the first song by The Chainsmokers

```
df[df['artist'] == 'The Chainsmokers']['title'].iloc[0]
```

Task 8: Which year is the song Closer by the The Chainsmokers produced

```
df[df['title'] == 'Closer']['year']
```

Name: _____

Date: _____

Lesson 3.2

3.2 Learning – Bar Graph, Line Graph

Write a Python Code to plot the Roblox revenue as a bar graph and line graph

Roblox revenue

Year	Revenue
2017	\$45 million
2018	\$335 million
2019	\$435 million
2020	\$920 million

Name: _____

Date: _____

Lesson 3.2

Create a **bar** plot for **Roblox revenue**

Step 1: Import the relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Step 2: Create the DataFrame

```
robloxRevenue = {
    'year': [2017, 2018, 2019, 2020],
    'revenue': [45, 335, 435, 920]
}

df = pd.DataFrame(robloxRevenue)
```

Step 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1,figsize=(10,5))
```

Step 4: Setting the width of bar & indexes of the bar on the x-axis

```
w = 0.2
x = np.asarray(df['year'])
```

Name: _____

Date: _____

Lesson 3.2

Step 5: Plot the bar graphs

```
ax1.bar(x, df['revenue'], width = w, color="Blue")
```

Step 6: Set title & x-label, y-label

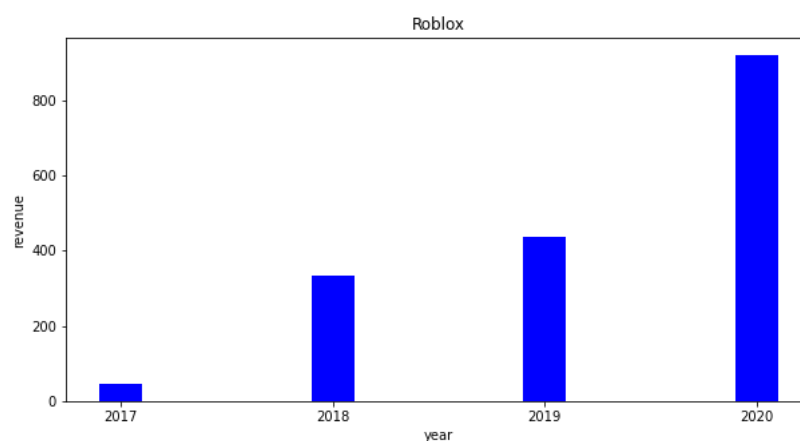
```
ax1.set_title("Roblox")  
ax1.set_xlabel("Year")  
ax1.set_ylabel("Revenue")
```

Step 7: Set the x ticks

```
ax1.set_xticks(x)
```

Step 8: Display the plot

```
plt.show()
```



Name: _____

Date: _____

Lesson 3.2

Create a **line** plot for **Roblox revenue**

Step 1: Import the libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Step 2: Load in DataFrame

```
robloxRevenue = [
    {'year': 2017, 'revenue': 45},
    {'year': 2018, 'revenue': 335},
    {'year': 2019, 'revenue': 435},
    {'year': 2020, 'revenue': 920},
]

df = pd.DataFrame(robloxRevenue)
```

Step 3: Create the figure and axes

```
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 5))
```

Step 4: Plot the bar graph

```
ax1.plot(df['year'], df['revenue'])
```


Name: _____

Date: _____

Lesson 3.2

Step 5: Set the title & x-label, y-label

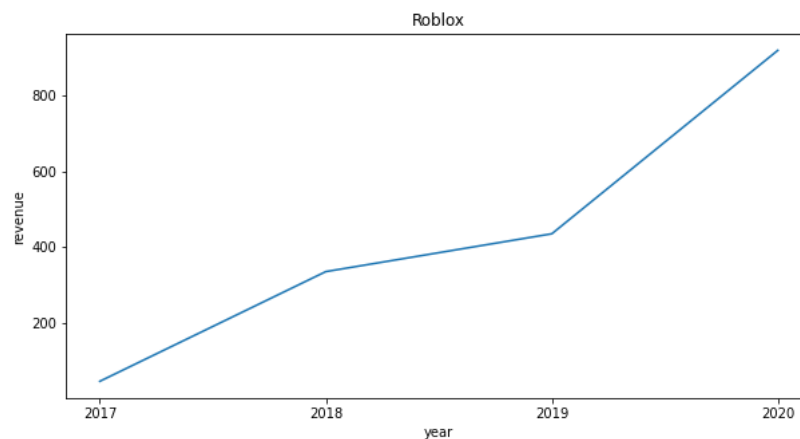
```
ax1.set_title("Roblox")  
ax1.set_xlabel("Year")  
ax1.set_ylabel("Revenue")
```

Step 6: Set x ticks

```
ax1.set_xticks(x)
```

Step 7: Display the graph

```
plt.show()
```



Name: _____

Date: _____

Lesson 3.2**3.2 Practice – Bar Graph, Line Graph**

Write a python code to plot a bar graph and line graph for Minecraft revenue and Candy Crush revenue respectively.

Minecraft revenue

Year	Revenue
2012	\$211 million
2013	\$326 million
2014	\$165 million
2015	\$350 million
2016	\$420 million
2017	\$370 million
2018	\$500 million
2019	\$375 million
2020	\$415 million

Candy Crush revenue

Year	Revenue
2012	\$77 million
2013	\$230 million
2014	\$1130 million
2015	\$1293 million
2016	\$784 million
2017	\$695 million
2018	\$930 million
2019	\$1117 million
2020	\$1190 million

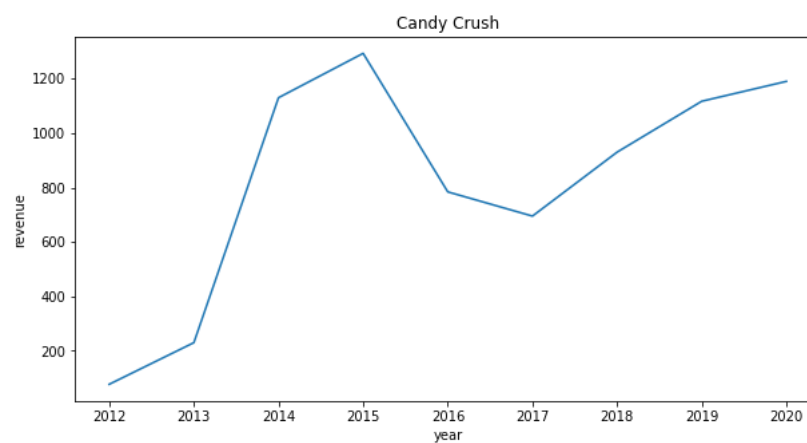
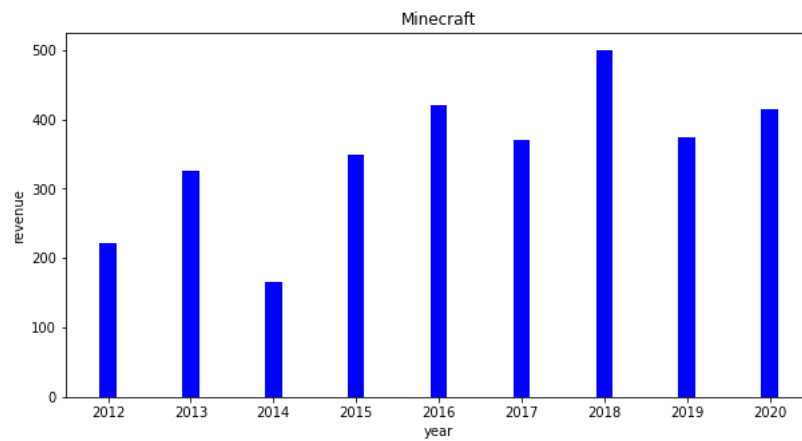
No Guided Solution Provided

Name: _____

Date: _____

Lesson 3.2

Expected Graphs



Name: _____

Date: _____

Lesson 3.3**3.3 Learning – Multiplot****Line Graph**

A	2	4	2	6	1	2	3	6	8
B	1	5	5	3	2	1	2	3	4

Bar Graph

	PlayersA	PlayersB
Player1	15	115
Player2	45	145
Player3	32	132

Name: _____

Date: _____

Lesson 3.3

Line Graph

A	2	4	2	6	1	2	3	6	8
B	1	5	5	3	2	1	2	3	4

Task 1: Import libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Task 2: Load in DataFrame

```
A = [2, 4, 2, 6, 1, 2, 3, 6, 8]
B = [1, 5, 5, 3, 2, 1, 2, 3, 4]
df = pd.DataFrame({'A':A, 'B':B})
```

Task 3: Create Figure & Axes

```
fig1,ax1= plt.subplots(1,1,figsize=(10,5))
```

Task 4: Plot the line graph

```
ax1.plot(df['A'], color="Green", label="A")
ax1.plot(df['B'], color="Red", label="B")
```

To overlay the different graphs on the same line plot, we will just need to plot and give them different labels

Name: _____

Date: _____

Lesson 3.2

Task 5: Set the title, x-label & y label

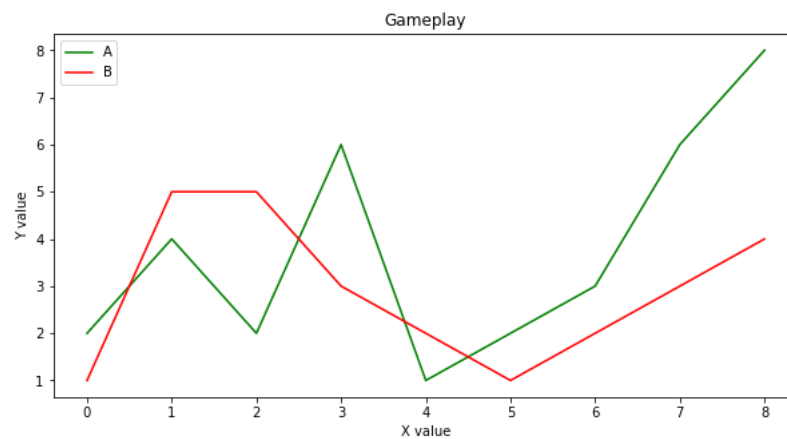
```
ax1.set_title("Gameplay")
ax1.set_xlabel("X value")
ax1.set_ylabel("Y value")
```

Task 6: Set x ticks

```
ax1.set_xticks(df.index)
```

Task 7: Display the plot & legend

```
plt.legend()
plt.show()
```



Name: _____

Date: _____

Lesson 3.3**Bar Graph**

	PlayersA	PlayersB
Player1	15	115
Player2	45	145
Player3	32	132

Task 1: Import relevant libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Task 2: Load DataFrame

```
playersA = [
    { "name": "player1", "hours": 15},
    { "name": "player2", "hours": 45},
    { "name": "player3", "hours": 32}
]

playersB = [
    { "name": "player1", "hours": 115},
    { "name": "player2", "hours": 145},
    { "name": "player3", "hours": 132}
]

df_A = pd.DataFrame(playersA)
df_B = pd.DataFrame(playersB)
```

Task 3: Create the figure and axes

```
fig1,ax1= plt.subplots(1,1,figsize=(10,5))
```

Name: _____

Date: _____

Lesson 3.3**Task 4:** Setting the width of bar & indexes of the bar on the x-axis

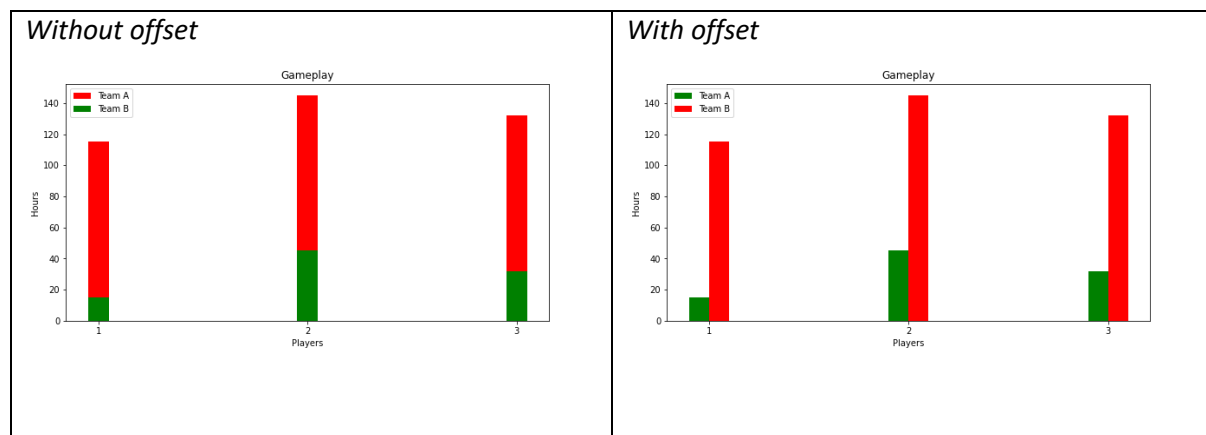
```
w = 0.1
x = np.arange([i+1 for i in range(len(df_A))])
```

w is not 1 because 1 means it occupies the entire space. It is only useful when we are doing single bar plots side by side like a histogram

Task 5: Plotting the bar graphs

```
ax1.bar(x - w/2, df_A['hours'], width = w, color="Green", label="Team A")
ax1.bar(x + w/2, df_B['hours'], width = w, color="Red", label="Team B")
```

x-w/2 & x+w/2 are essentially the offset. If we were to use x for both, we are overlaying them directly on each other which is not what we want to display. Therefore, +w/2 and -w/2 helps to shift the bar to the left or right of each other



Name: _____

Date: _____

Lesson 3.3

`matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)`

How can I effectively have multiple bars graph on the same plot?

We will need to understand the parameters when plotting bar

1. *x*
 - *The x coordinates of the bars*
2. *height*
 - *The height(s) of the bars*
3. *Width*
 - *The width(s) of the bars*

Normally, you will notice that we do

```
w = 0.2
x = np.asarray([i for i in range(len(df_A))])
ax1.bar(x, df_A['hours'], width = w)
```

Width will indicate how wide it is, left to right

When we want multiplot, we do not want to have them all align to specific x coordinate otherwise it will overlay each other. That is why

```
w = 0.2
x = np.asarray([i for i in range(len(df_A))])
ax1.bar(x-0.1, df_A['hours'], width = w)
ax1.bar(x+0.1, df_B['hours'], width = w)
```

$x - 0.1$ means shift to the left by 0.1

$x + 0.1$ means shift to the right by 0.1

Name: _____

Date: _____

Lesson 3.3

Task 6: Set the title, x-label & y-label

```
ax1.set_title("Gameplay")
ax1.set_xlabel("Players")
ax1.set_ylabel("Hours")
```

Task 7: Set x-ticks

```
ax1.set_xticks(x)
```

Task 8: Display the plot

```
plt.legend()
plt.show()
```

Name: _____

Date: _____

Lesson 3.3**3.3 Practice – Multiplot**

Year	Roblox	Minecraft	Candy Crush
2012		211	77
2013		326	230
2014		165	1130
2015		350	1293
2016		420	784
2017	45	370	695
2018	335	500	930
2019	435	375	117
2020	920	415	1190

Write a Python Code using Matplotlib to create a multiplot for Roblox, Minecraft, Candy Crush in line graphs & bar graphs.

Do note that for the bar graph, we will need to consider the offset and width to prevent overlaps.

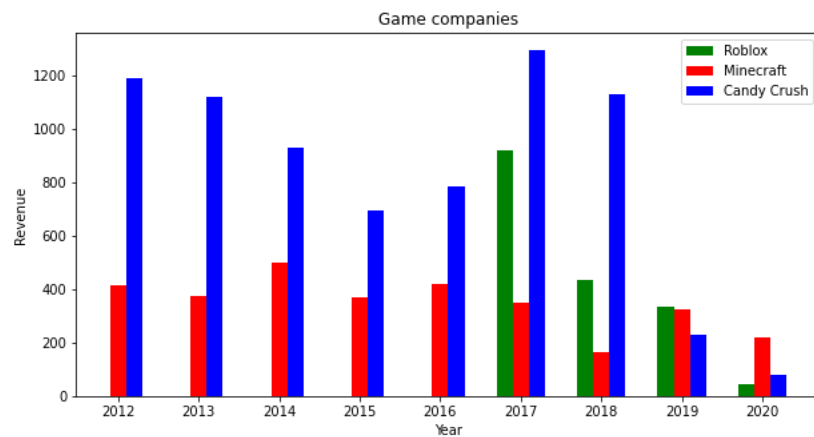
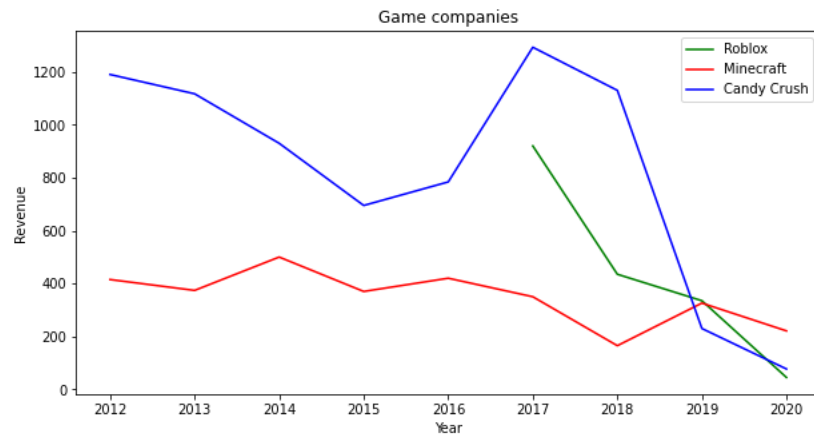
No Guided Solution Provided

Name: _____

Date: _____

Lesson 3.3

Expected Graphs



Name: _____

Date: _____

Lesson 3.3

Getting the data

```
# Task 1: Import the relevant libraries
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Task 2: Create DataFrame

robloxRevenue = {
    'year': [2017, 2018, 2019, 2020],
    'revenue': [45, 335, 435, 920]
}

minecraftRevenue = {
    'year': [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
    'revenue': [221, 326, 165, 350, 420, 370, 500, 374, 415]
}

candycrushRevenue = {
    'year': [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
    'revenue': [77, 230, 1130, 1293, 784, 695, 930, 1117, 1190]
}

df_A = pd.DataFrame(robloxRevenue)
df_B = pd.DataFrame(minecraftRevenue)
df_C = pd.DataFrame(candycrushRevenue)
```

Name: _____

Date: _____

Lesson 3.3**Line Graph**

```
# Step 3: Create the figure and axes
fig1,ax1= plt.subplots(1,1, figsize=(10,5))

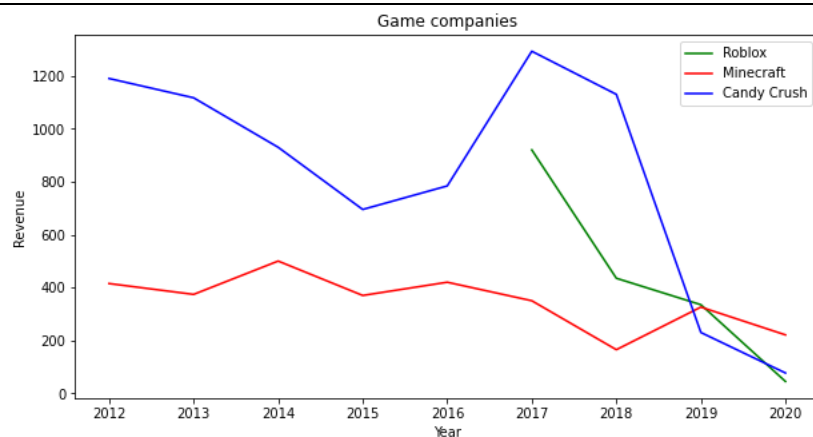
# Step 4: Setting indexes of each plot on the x-axis
x_A = np.asarray([max(df_C['year'])-i for i in range(len(df_A))])
x_B = np.asarray([max(df_C['year'])-i for i in range(len(df_B))])
x_C = np.asarray([max(df_C['year'])-i for i in range(len(df_C))])

# Step 5: Plot the bar graphs
ax1.plot(x_A, df_A['revenue'], color="Green", label="Roblox")
ax1.plot(x_B, df_B['revenue'], color="Red", label="Minecraft")
ax1.plot(x_C, df_C['revenue'], color="Blue", label="Candy Crush")

# Step 6: Set the title, x-label & y-label
ax1.set_title("Game companies")
ax1.set_xlabel("Year")
ax1.set_xticks(df_C['year'])

# Step 7: Set x-ticks
ax1.set_xticks(x_C)

# Step 8: Display the plot
plt.legend()
plt.show()
```



Name: _____

Date: _____

Lesson 3.3**Bar Graph**

```

# Step 3: Create the figure and axes
fig1,ax1= plt.subplots(1,1, figsize=(10,5))

# Step 4: Setting the width of bar & indexes of the bar on the x-axis
w = 0.2
x_A = np.asarray([max(df_C['year'])-i for i in range(len(df_A))])
x_B = np.asarray([max(df_C['year'])-i for i in range(len(df_B))])
x_C = np.asarray([max(df_C['year'])-i for i in range(len(df_C))])

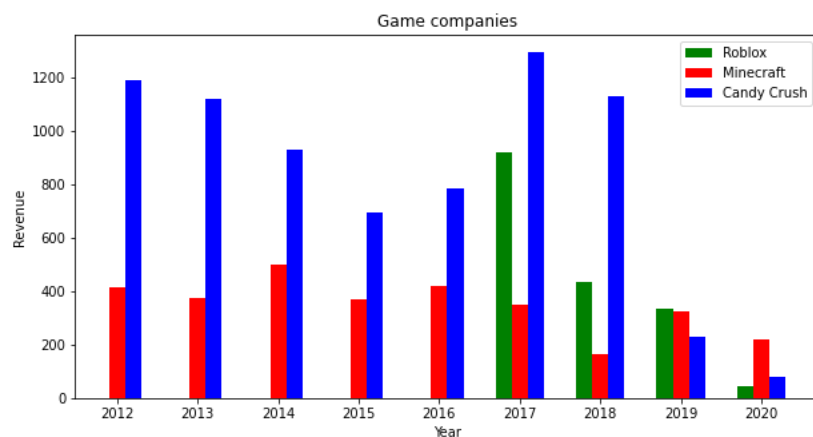
# Step 5: Plot the bar graphs
ax1.bar(x_A-w, df_A['revenue'], width=w, color="Green", label="Roblox")
ax1.bar(x_B+0, df_B['revenue'], width=w, color="Red", label="Minecraft")
ax1.bar(x_C+w, df_C['revenue'], width=w, color="Blue", label="Candy Crush")

# Step 6: Set the title, x-label & y-label
ax1.set_title("Game companies")
ax1.set_xlabel("Year")
ax1.set_ylabel("Revenue")

# Step 7: Set x-ticks
ax1.set_xticks(df_C['year'])

# Step 8: Display the plot
plt.legend()
plt.show()

```



Name: _____

Date: _____

Day 4

	Lesson Plan
4	- Data Analysis: % of o level cohort that progressed to post-secondary education

**Note: The CSV files obtained are from the data.gov.sg source and it is clean, therefore data cleaning is not required.*

<https://data.gov.sg/dataset/percentage-of-o-level-cohort-that-progressed-to-post-secondary-education>

Name: _____

Date: _____

Lesson 4

**Note: The CSV files obtained are from the data.gov.sg source and it is clean, therefore data cleaning is not required.*

Exercise 1: Finding out the percentage of o level cohort that progressed to post-secondary education

<https://data.gov.sg/dataset/percentage-of-o-level-cohort-that-progressed-to-post-secondary-education>

Exercise 1a:

Task 1a:

Let's analyse the data from the CSV files using the following:

Write the output/description in the space below.

`df.head()`

`df.tail()`

`df.info()`

`df.describe()`

`df.columns`

In your own words, explain what the data is about

Which type of graph do we use?

Name: _____

Date: _____

Lesson 4

Task 1a:

Line Plotting: For the below data, we are required to do one plot, with 2 graphs

Graph 1 - Percentage of Malay who progressed to post-secondary over the years 2008 to 2020

Graph 2 - Percentage of Indian who progressed to post-secondary over the years 2008 to 2020

Task 2: Do we have an idea of what is in the CSV file? Keep in mind our goal for the plot

1 line graph of Year (x-axis) vs Malay percentage (y-axis)

1 line graph of Year (x-axis) vs Indian percentage (y-axis)

Let's start the code step by step:

Step 1:

```
import matplotlib.pyplot as plt
import pandas as pd
```

Step 2:

#Create 1 Pandas DataFrame for the entire CSV file.

2	<code>df=pd.read_csv('dataname')</code>
Ans	<code>df = pd.read_csv('percentage-of-o-level-cohort-that-progressed-to-post-secondary-education.csv')</code> <code>df</code>

Step 3:

#Remove NA and "-" for value columns

3	<code>df=df[df['columnname']!= 'na']</code>
Ans	<code>df = df[df['percentage_progress_postsec'] != 'na']</code>
Ans	<code>df = df[df[' percentage_progress_postsec'] != '-']</code>

Name: _____

Date: _____

Lesson 4

Step 4:

#Change all columns you would like to plot in the graph with “numerical values” to numerics
(they are not in integers format)

4	<code>df[“columnname”]=pd.to_numeric(df[“columnname”])</code>
Ans	<code>df['percentage_progress_postsec'] = pd.to_numeric(df['percentage_progress_postsec'])</code>
Ans	<code>df['year'] = pd.to_numeric(df['year'])</code>

Let's go back to our goal for the plot:

1 line graph of Year (x-axis) vs Malay percentage (y-axis)

1 line graph of Year (x-axis) vs Indian percentage (y-axis)

What is all the data in our CSV file? How do we get the data for our x and y axis?

We have all the races – Chinese, Indian, Malay, Others.

So What do we need to do? Filter and Extract to get to our goal for the race for each of the year 2008 till 2020

Step 5: Filter and extract the Malay data and Indian data separately

5	<code>df_newname = df_oldname[(df_oldname[“columnname”] == criteria & (df_oldname[“columnname”]==criteria))]</code>
Ans	<code>df_indian = df[(df['year'] >= 2008) & (df['year'] <= 2020)]</code> <code>df_indian = df[(df['race'] == "Indian")]</code>
Ans	<code>df_malay = df[(df['year'] >= 2008) & (df['year'] <= 2020)]</code> <code>df_malay = df[(df['race'] == "Malay")]</code>

Name: _____

Date: _____

Lesson 4

Step 6: Extract the columns required

6	<pre>df_newname = df_newname[["columnname","columnname"]] df_newname= df_newname.sort_values(by= "columnname")</pre>
Ans	<pre>df_malay = df_malay[['year', 'percentage_progress_postsec']] df_malay = df_malay.sort_values (by='year')</pre>
Ans	<pre>df_indian = df_indian[['year', 'percentage_progress_postsec']] df_indian = df_indian.sort_values (by='year')</pre>

Step 7: Remove duplicates

7	<pre>df=df.drop_duplicates()</pre>
Ans	<pre>df = df.drop_duplicates()</pre>

Step 8: Plot line graph

9	<pre>fig1.ax1=plt.subplots(1,1,figsize=(10,5)) ax1.plot(df."columnname" , df."columnname", label= "labelname")</pre>
Ans	<pre>fig1.ax1=plt.subplots(1,1,figsize(1,1)) ax1.plot(df_malay.year, df_malay.percentage_progress_postsec, label="Malay") ax1.plot(df_indian.year, df_indian.percentage_progress_postsec , label="Indian")</pre>

Name: _____

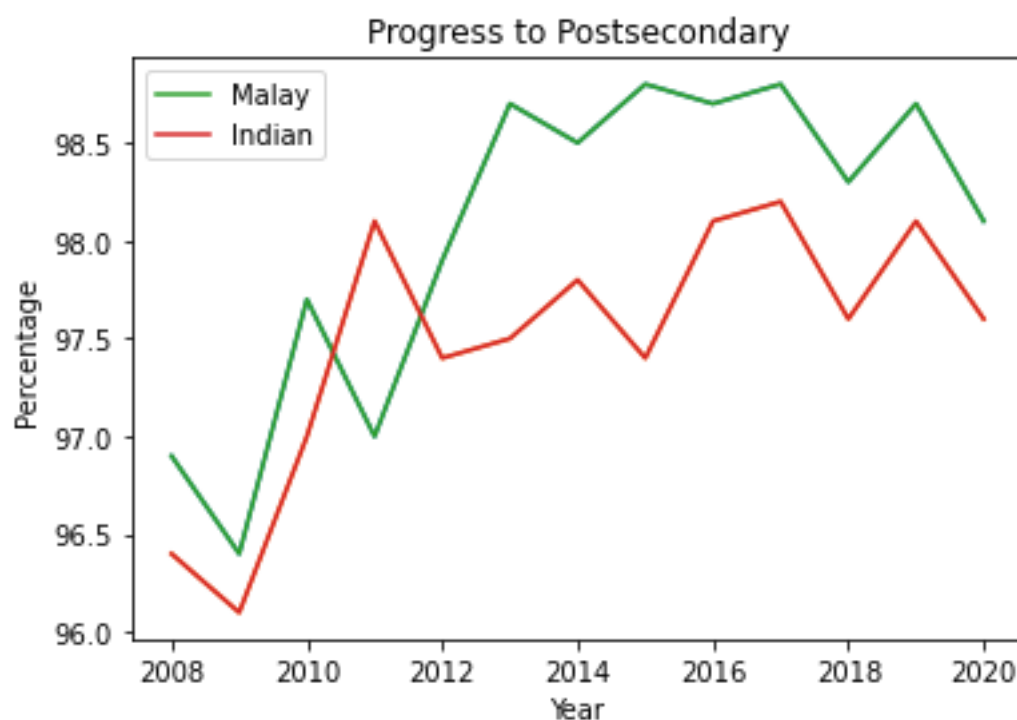
Date: _____

Lesson 4

Step 9: Display plot with labels

8	<pre>ax1.set_xlabel("columnname") ax1.set_ylabel("columnname") ax1.set_title("titlename")</pre>
Ans	<pre>ax1.set_xlabel("Year") ax1.set_ylabel("Percentage") ax1.set_title("Progress to Postsecondary") plt.legend() plt.show()</pre>

Output:



Python

Data Analytics Course – Dataset 1



Name: _____

Date: _____

Lesson 4

Exercise 1a: Data Analysis:

Data Analysis Functions:

get mean

1	df["dataname"].mean()
---	-----------------------

get median

1	df["dataname"].median()
---	-------------------------

#indexing for extracting data

1	df['columntomanipulate'][index]
---	---------------------------------

#finding max

1	df["dataname"].max()
---	----------------------

#finding min

1	df["dataname"].min()
---	----------------------

Name: _____

Date: _____

Lesson 4

Exercise 1a: Data Analysis:

Do the following analysis:

Question 1: What is the mean percentage of Malay students that progress up to post-secondary?

Question 2: What is the median percentage of Indian students that progress up to post-secondary?

Question 3: Which is the year where highest percentage of Indian students progressed up to post-secondary and what is that percentage?

Question 4: Which is the year where lowest percentage of Malay students progressed up to post-secondary and what is that percentage?

Exercise 1b: Plotting 2 bar graphs side by side in 1 plot

- i) Using the same data, plot 2 bar graphs side by side to compare the percentage of Chinese and Others who progressed up to secondary education from 2008 to 2020
- ii) Do an short analysis of:
 - Mean percentage of each race that progress up to post- secondary
 - Years where Others percentage is higher than Chinese

Name: _____

Date: _____

Lesson 4

Exercise 1b

Step 1:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Step 2:

#Create 1 Pandas DataFrame for the entire CSV file.

2	<code>df=pd.read_csv('dataname')</code>
Ans	<code>df = pd.read_csv('percentage-of-o-level-cohort-that-progressed-to-post-secondary-education.csv')</code> <code>df</code>

Step 3:

#Remove NA and "-"for value columns

3	<code>df=df[df['columnname']!= 'na']</code>
Ans	<code>df = df[df['percentage_progress_postsec'] != 'na']</code>
Ans	<code>df = df[df[' percentage_progress_postsec'] != '-']</code>

Name: _____

Date: _____

Lesson 4

Step 4:

#Change all columns you would like to plot in the graph with “numerical values” to numerics
(they are not in integers format)

4	<code>df[“columnname”]=pd.to_numeric(df[“columnname”])</code>
Ans	<code>df['percentage_progress_postsec'] = pd.to_numeric(df['percentage_progress_postsec'])</code>
Ans	<code>df['year'] = pd.to_numeric(df['year'])</code>

Let's go back to our goal for the plot:

1 bar graph of Year (x-axis) vs Chinese percentage (y-axis)

beside

1 bar graph of Year (x-axis) vs Others percentage (y-axis)

What is all the data in our CSV file? How do we get the data for our x and y axis?

We have all the races – Chinese, Indian, Malay, Others.

So What do we need to do? Filter and Extract to get to our goal for the race for each of the year 2008 till 2020

Step 5: Filter and extract the Chinese data and Others data separately

5	<code>df_newname = df_oldname[(df_oldname[“columnname”] == criteria & (df_oldname[“columnname”]==criteria))]</code>
Ans	<code>df_chinese = df[(df['year'] >= 2008) & (df['year'] <= 2020)]</code> <code>df_chinese = df[(df['race'] == "Chinese")]</code>
Ans	<code>df_others = df[(df['year'] >= 2008) & (df['year'] <= 2020)]</code> <code>df_others = df[(df['race'] == "Others")]</code>

Name: _____

Date: _____

Lesson 4

Step 6: Extract the columns required

6	<pre>df_newname = df_newname[["columnname","columnname"]] df_newname= df_newname.sort_values(by= "columnname")</pre>
Ans	<pre>df_chinese = df_chinese[['year', 'percentage_progress_postsec']] df_chinese = df_chinese.sort_values (by='year')</pre>
Ans	<pre>df_others = df_others[['year', 'percentage_progress_postsec']] df_others = df_others.sort_values (by='year')</pre>

Step 7: Remove duplicates

7	<pre>df=df.drop_duplicates()</pre>
Ans	<pre>df = df.drop_duplicates()</pre>

*Step 8a: Define x position for 2 bar graphs

7	<pre>x=np.arange(len(dataframename))</pre>
Ans	<pre>x_axis= np.arange (len(df_others))</pre>

Name: _____

Date: _____

Lesson 4**Step 8b: Plot 2 bar graphs in one plot**

9	<pre>fig1.ax1=plt.subplots(1,1,figsize=(10,5)) ax1.bar(x-0.2, dataframecolumn,0.4,label="labelname") ax1.bar(x+0.2, dataframecolumn,0.4,label="labelname")</pre>
Ans	<pre>ax1.bar(x_axis- 0.2, df_chinese.percentage_progress_postsec,0.4,label="Chinese ") ax1.bar(x_axis+0.2, df_others.percentage_progress_postsec,0.4, label="Others")</pre>

Note:

0.2 is a number that has to be determined via trial and error depending on the amount of graphs you have.

0.4 is also a number that has to be determined via trial and error depending on the amount of graphs you have.

Step 9: Display plot with labels

8	<pre>ax1.set_xlabel("columnname") ax1.set_ylabel("columnname") ax1.set_title("titlename") plt.xticks(x,dataframecolumn)</pre>
Ans	<pre>ax1.set_xlabel("Year") ax1.set_ylabel("Percentage") ax1.set_title("Progress to Postsecondary") plt.xticks(x_axis,df['year']) plt.legend() plt.show()</pre>

Name: _____

Date: _____

Lesson 4

(Independent Exercise)

Exercise 2: Using the same data as above to find out the percentage of o level cohort that progressed to post-secondary education

<https://data.gov.sg/dataset/percentage-of-o-level-cohort-that-progressed-to-post-secondary-education>

Note: Be sure to annotate each step to getting your graph and conclusion

2a. Plot 1 line graph in one plot to compare the percentage of Chinese who progressed to post-secondary over the years 2008 to 2020

Do the following analysis:

Question 1: Which year was the lowest percentage of Chinese who progressed to post-secondary

Question 2: Is the percentage an upward trend, downward trend? Any specific pattern?

2b. Plot 1 bar graph in one plot to compare the percentage of Indians who progressed to post-secondary over the years 2008 to 2020.

Do the following analysis:

Question 1: Which year was the lowest and highest percentage of Indians who progressed to post-secondary

2c. Plot 4 bar graphs in 1 plot to compare the percentage of Malay, Chinese, Indian and Others who progressed to post-secondary for the years 2018, 2019 and 2020

Do the following analysis:

Question 1: Which race was consistently the highest over the 3 years?

Question 2: Which race saw an improvement in percentage over the 3 years?

Question 3: Which race saw a drop in percentage over the 3 years?
