DAILY ASSESSMENT FORMAT

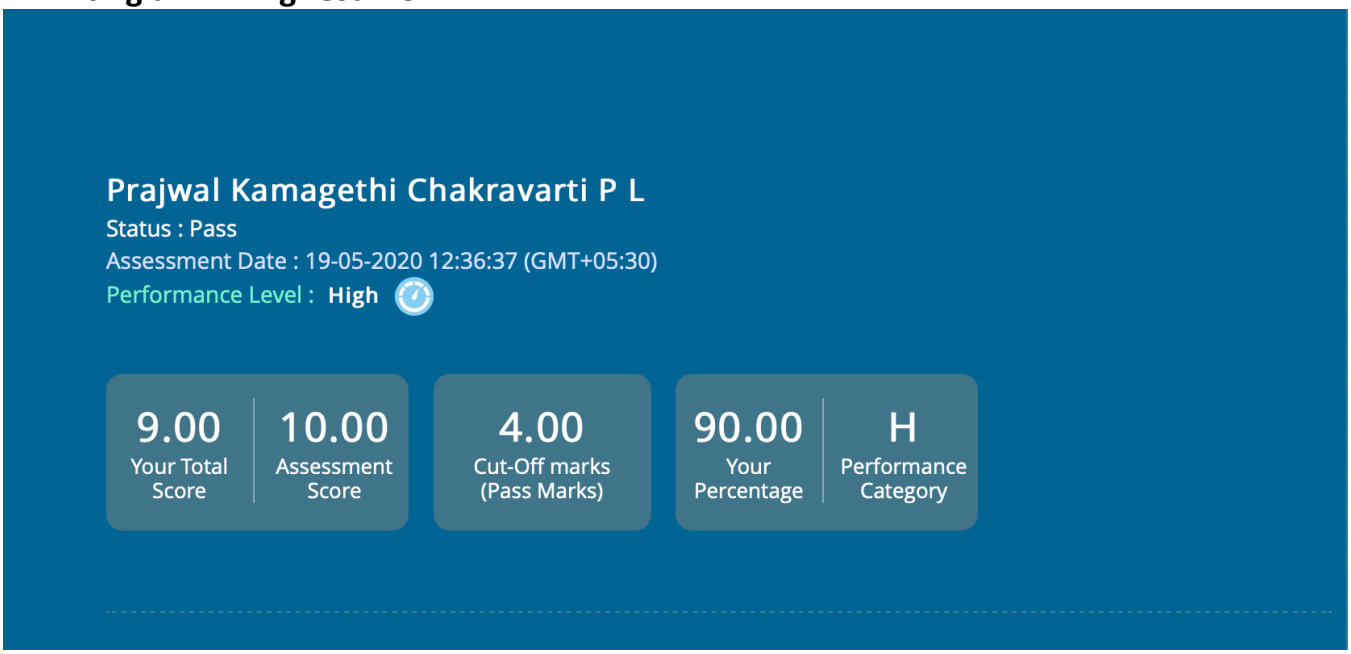
Date:	**19/05/2020**	**Name:**	**Prajwal Kamagethi Chakravarti P L**
Course:	**TCSion**	**USN:**	**4AL17EC073**
Topic:	**1.Gain guidance from career gurus 2.Writing a winning resume 3. Stay ahead in group discussion**	**Semester & Section:**	**6 B**
Github Repository:	**https://github.com/alvas-education-foundation/Prajwal-Kamagethi.git**		

FORENOON SESSION DETAILS

Report – Report can be typed or hand written for up to two pages.

1.Gain guidance from career gurus

2.Writing a winning resume



3. Stay ahead in group discussion

Prajwal Kamagethi Chakravarti P L

Status : Pass

Assessment Date : 19-05-2020 13:03:17 (GMT+05:30)

Performance Level : **Excellent**

10.00	10.00	4.00	100.00	E
Your Total Score	Assessment Score	Cut-Off marks (Pass Marks)	Your Percentage	Performance Category

HEADSTART:

- **Learnt why we need a head start.**
- **Intensive competition**
- **Talent Acquisition**
- **Employable skills**
- **Changing job roles**
- **Employment outlook – positive.**

6 Key pillars to get a head start:

- **Clarity of thoughts**
- **Access and visibility**
- **Early Preparation**
- **Acquire relevant skills**
- **Compelling resume**
- **Cracking the interview.**

Resume and cover letter:

- **The Resume should be crisp and to the point**
- **The resume should be clear about your carrier objectives, skills, abilities and what you are looking for.**
- **Choose the format or style that is best suited for your profile.**
- **Do not lie in the resume.**
- **Be proud of your achievements and experience as the resume is YOU on paper.**
- **A cover letter gives initial impression of you.**
- **A cover letter expresses points that your resume might not cover.**

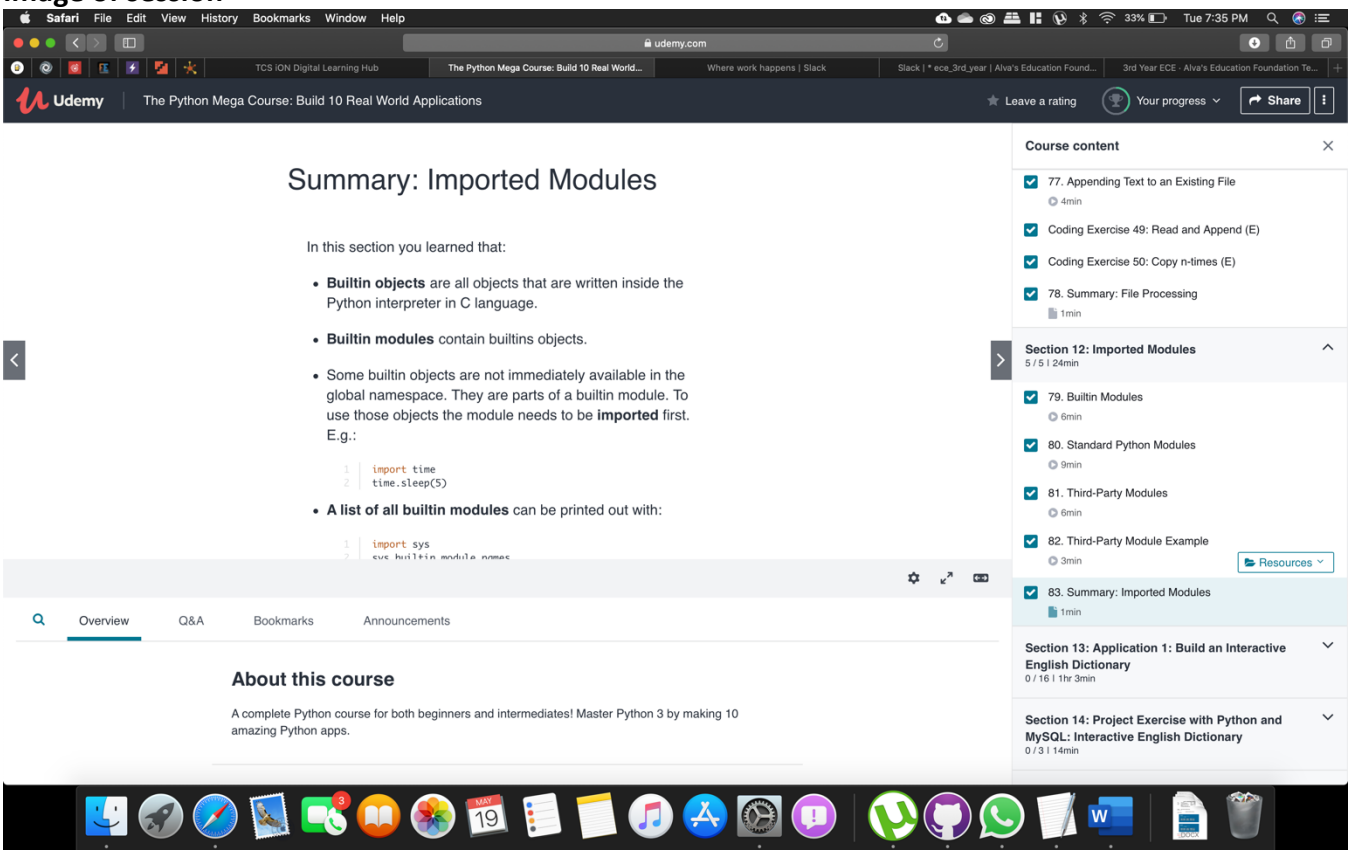
Group discussion:

- **A group discussion is not a debate.**
- **To be aware of your body language.**
- **To keep a check on your tone of voice and the language used.**
- **To update yourself with current information.**
- **Not to panic.**
- **To maintain formal decorum.**

Date:	**19/05/2020**	**Name:**	**Prajwal Kamagethi Chakravarti P L**
Course:	**Python**	**USN:**	**4AL17EC073**
Topic:	The Basics: Processing User Input, The Basics: Loops, Putting the Pieces Together: Building a Program, List Comprehensions, More on Functions, File Processing	**Semester & Section:**	**6 B**
Github Repository:	**https://github.com/alvas-education-foundation/Prajwal-Kamagethi.git**		

AFTERNOON SESSION DETAILS

Image of session



Processing User Input:

- A Python program can get **user input** via the `input` function:
- The **input function** halts the execution of the program and gets text input from the user**:**
- The input function converts any **input to a string**, but you can convert it back to int or float:

```
experience_months = input("Enter your experience in months: ")

experience_years = int(experience_months) / 12
```

- You can **format strings** with (works both on Python 2 and 3):

```
name = "Sim"

experience_years = 1.5

print("Hi %s, you have %s years of experience." % (name, experience_years))
```

Output: `Hi Sim, you have 1.5 years of experience.`

- You can also **format strings** with (Python 3 only):

```
name = "Sim"

experience_years = 1.5

Print("Hi {}, you have {} years of experience".format(name, experience_years))
```

Loops:

- **For loops** are useful for executing a command over a large number of items.

- You can loop over **dictionary keys**:

```
phone_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"}

for value in phone_numbers.keys():

 print(value)
```

- You can loop over **dictionary values**:

```
phone_numbers = {"John Smith":"+37682929928","Marry Simpons":"+423998200919"}

for value in phone_numbers.values():

print(value)
```

List Comprehensions:

- A list comprehension is an expression that creates a list by iterating over another container.
- A **basic** list comprehension:
  - [i*2 for i in [1, 5, 10]]
- List comprehension with **if** condition:
  - [i*2 for i in [1, -2, 10] if i>0]
- List comprehension with an **if and else** condition:
  - [i*2 if i>0 else 0 for i in [1, -2, 10]]

More on Functions:

- Functions can have more than one **parameter**:
- Functions can have **default** parameters
- Arguments can be passed as **non-keyword** (positional) arguments or **keyword** arguments
- An ***args** parameter allows the function to be called with an arbitrary number of non-keyword arguments
- An ****kwargs** parameter allows the function to be called with an arbitrary number of keyword arguments.

File Processing:

- You can **read** an existing file with Python:

```
with open("file.txt") as file:
    content = file.read()
```

- You can **create** a new file with Python and **write** some text on it:

```
with open("file.txt", "w") as file:
    content = file.write("Sample text")
```

- You can **append** text to an existing file without overwriting it:

```
with open("file.txt", "a") as file:
    content = file.write("More sample text")
```

- You can both **append and read** a file with:

```
with open("file.txt", "a+") as file:
    content = file.write("Even more sample text")
    file.seek(0)
    content = file.read()
```

- **Builtin objects** are all objects that are written inside the Python interpreter in C language.

- **Builtin modules** contain builtins objects.

- Some builtin objects are not immediately available in the global namespace. They are parts of a builtin module. To use those objects the module needs to be **imported** first. E.g.:

  - import time
  - time.sleep(5)
- **A list of all builtin modules** can be printed out with:

  - import sys
  - sys.builtin_module_names
- **Standard libraries** is a jargon that includes both builtin modules written in C and also modules written in Python.

- **Standard libraries** written in Python reside in the Python installation directory as *.py* files. You can find their directory path with `sys.prefix`.

- **Packages** are a collection of *.py* modules.

- **Third-party libraries** are packages or modules written by third-party persons (not the Python core development team).

- Third-party libraries can be **installed** from the terminal/command line:

- Mac and Linux:

- `pip3 install pandas` or use `python3 -m pip install pandas` if that doesn't work.