# Sections 3 and 4

**1. How to create a Python file in Pycharm tool?**

Right-click on Project --> New --> Python File--> Enter File Name

**2. How to print a text in Pycharm console?**

```
print("Hello")
```

**3. How to execute Python script?**

Right-click on the Python file --->Run 'File Name'

**4. Shortcut keys to execute Python script?**

CTRL + SHIFT + F10

**5. How to comment a line in Python file?**

Using # at the begining of the line

**6. How to define a variable?**

```
a = 3
a = "Python"
```

Note: No need to explicitly declare data types

**7. How to print 2 variables of different data types in Python?**

```
b=3
print("{} {}".format("Value is",b))
```

**Note:** Each curly brace will store the value of one variable. In this example 'Value is' is stored in first {} and value of b in second {}

**8. How to print datatype of the variable?**

Using type keyword

```
b=3
print(type(b))
```

**9. How to declare list?**

Using square brackets[]

```
a=[1, 2, 3, "Rahul"]
```

**10. How to insert value in the list?**

Using insert keyword

```
a=[1, 2, 3, "Rahul"]
a.insert(1, "Hello")
print(a)
```

Output: [1, Hello, 2, 3, 'Rahul']

**11. How to append value in the list**
Using append keyword
a=[1, 2, 3, "Rahul"]
a.append("End")
print(a)

Output: [1, 2, 3, 'Rahul','End']

**12. How to delete a value from the list?**
**Using del keyword**
a=[1, 2, 3, "Rahul"]
del a[0]
print(a)

Output: [2, 3, 'Rahul']

**13. How to declare tuple?**
Using circular braces ()
a=(1, 2, 3)
print(a)

Output - [1, 2, 3]

**Note**: Tuple is immutable(it means we cannot re-assign the value)

14. **How to declare dictiionary?**
It is declared in key and value pair form
a = {1: "first name", 2: "second name", "age":25}
print(a["age"])
print(a[0])

Output -
25
first name

**Python Data Types reference** - https://www.digitalocean.com/community/tutorials/python-data-types

# Section 5:

1. How to use If else condition?

```
greeting = "Hello, Good Morning!"

if greeting == "Hello, Good Morning!":
        print("Condition matches")
else:
        print("Condition do not match")
```

Note:
- if else is a condition
- Colon(:) is used to indicate the start of a new block of code
- Code indentation is important for python to understand the code format.

2. How to use For loop?

```
listOfNumber = [1, 2, 5, 6, 8]
for i in listOfNumber:
        print(i)
```

Note:
- for is loop which helps to iterate over the list, tuple or dictionary, String etc.
- Variable 'I' used in the above example will help to iterate through the list
- Colon(:) is used to indicate the start of a new block of code
- Code indentation is important for python to understand the code format.

3. How to use while loop?

```
it = 10

while it > 5:
        print(it)
        it = it - 1
print(a)
```

Note:
- while loop while continue iterating until the condition is false
- There are 2 keywords that can be used with while loop – break and continue
- break keyword – it will terminate the entire loop
-continue keyword – It will terminate the current execution of the loop and will start from top of the loop
- Colon(:) is used to indicate the start of a new block of code
- Code indentation is important for python to understand the code format.

4. What is function and how to define a function?
- Function is a group of related statements that perform a specific task

```
def greetMe():
        print("Good morning!")
greetMe()
```

**-def is used to define a method/function**

**- Just after def you define function name**

**- Colon(:) is used to indicate the start of a new block of code**

**- Execute the function by calling the function name – greetMe()**

# Section 6:

1. **What is class?**
   - Class is a user defined blueprint or prototype
   - Class includes – Methods, Class variables, Instance variables, Constructor, etc.

2. **Below examples will help us to understand,**
   **-** Defining a class, variable and method
   - Creating class object
   - Calling method using class object

```
class NewClass:
        num = 190       # Class variable

        def getData(self):     # Method declaration with self which is
auto populated
                print("I am executing method in the class")


obj = NewClass()     # Creating class-NewClass object
obj.getData()          # Calling method-getData() using class object-obj
print(obj.num)        # Printing class variable using class object-obj
```

3. **What is constructor?**
   - Constructor is a method which is called when the object of the class is created.
   - There are 2 types of constructor,

   Default Constructor: It is a default constructor which is called when there is no user defined constructor. It doesn't accept any arguments

   Parameterized Constructor: Constructor with parameters is known as parameterized constructor

**Example:**

```
class ConstTest: # class created
        num = 20          # class variable

        def __init__(self, a, b):      #Constructor is defined with __init__
keyword and self is the object which is initialized at the time of Object
creation
                self.firstNumber = a        # Assigning value 'a' to variable
'firstNumber'(Instance variable) with self which is mandatory
                self.secondNumber = b      # Assigning value 'b' to variable
'SecondNumber'(Instance variable) with self which is mandatory

        def summation(self):                # Method
                return self.firstNumber + self.secondNumber + self.num
# Calling Instance and Class variable using self


obj = ConstTest(50, 50)          # Creating class object and passing 2
variables because Constructor is a parameterized constructor with 2 variables
print(obj.summation())            # Calling method
```

4. **What is Inheritance?**
   - Accessing properties of all the Parent class in the Child class

```
# ConstTest is Parent class
# Parent class properties can be inherited in the below code format.

class ChildImpl(ConstTest):
```

5. **What is String?**
   - In Python, a string is a sequence of characters
6. **How to define a String?**

```
name = "Rahul"
```

7. **How to print a String?**

```
name = "Rahul"
print(name)
```

8. **How to print 4th character from the String**

```
name = "QAClick"
print(name[3])
```

9. **How to print substring from a String?**

```
name = "QAClick"
print(name[0:4])    # Print substring 0 to n-1
```

10. **How to check a String contains another String?**

```
name = "Rahul Shetty"
nameContains = "Shetty"
print(nameContains in name)          #To verify string contains another
string
```

11. **How to concatenate 2 strings?**

```
name = "Rahul Shetty"
greet = " Hello!"
print(greet+name)
```

12. **How to Trim space(s) from String?**

```
greet = " Hello! "
print(greet.strip())                #Trim begining and ending spaces
print(greet.rstrip())              #Trim ending spaces
print(greet.lstrip())              #Trim begining spaces
```

# Section 7:

1. **How to read characters from a text file?**

```
file = open('Content.txt')            #Open File
print(file.read())    #Read file
```

2. **How to read first and second line from the text file?**

```
print(file.readline())    #Read first line
print(file.readline())    #Read second line
```

3. **How to read and print line by line from a text file?**

```
file = open('Content.txt')
for line in file.readlines():
print(line)

file.close()              #Close the open file
```

4. **How to open the file in read mode?**

```
with open('Content.txt', 'r') as reader: #reader is a variable and 'r'
indicates file to be open in read mode
```

5. **How to open the file in write mode?**

```
with open('Content.txt', 'w') as writer: #writer is a variable and 'w'
indicates file to be open in write mode
```

6. **Exercise:** Read the file, store the values in list, reverse the list and write it in the same file.

```
with open('Content.txt', 'r') as reader:
        content = reader.readlines() #Store the values in content
        print(content)

with open('Content.txt', 'w') as writer:
        for i in reversed(content): #reversed(content) will reverse the
values
                writer.write(i) #write the values in the file
```

# Section 8:

1. **How to raise an exception?**
   -Using raise Exception keyword
   -Using assert keyword

#Using raise Exception keyword:

```
ItemsInCart = 0

# Exception with raise keyword : If the condition is not satisfied then it
will fail the case and raise the exception
if ItemsInCart != 2:
        raise Exception("Items in cart is not equals to 2")
```

#Using assert keyword:

```
ItemsInCart = 0
```

```
assert (ItemsInCart == 1)
```

2. **What is try except block and how it is used?**
   **- try block:** lets you test a block of code for errors. If an exception is raised, it jumps straight into the except block.
   **- except block:** this code is only executed if an exception occured in the try block. The except block is required with a try block, even if it contains only the pass statement.

```
#In this example, the filelog.txt file is missing which will raise an
exception and will be caught in except block
try:
        with open('filelog.txt', 'r') as reader:
                reader.read()
except:
        print("File not present")
```

3. **What is finally bock?**
   **-** The finally block runs whether or not the try statement produces an exception.

```
# finallly block is used with try and except block. finally block will
always execute.
try:
        with open('Content.txt', 'r') as reader:
                reader.read()
except Exception as e:
        print(e)
finally:
        print("Test Pass: Clean up process")
```

# Section 9:

1. **How to install selenium package in python interpreter?**
   **- Windows: File → Settings → Project Interpreter → Click + icon → Install Selenium**
   **- MAC OS: Preferences → Project Interpreter → Click + icon → Install Selenium**

2. **How to invoke Chrome browser using selenium webdriver?**
   **Latest selenium package:**

```
from selenium import webdriver
```

```
driver = webdriver.Chrome()
driver.get("https://rahulshettyacademy.com/loginpagePractise/")
```

Old selenium package:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

service_obj = Service("path\to\chromedriver.exe")
driver = webdriver.Chrome(service=service_obj)

driver.get("https://rahulshettyacademy.com/loginpagePractise/")
```

3. How to maximize browser window using selenium webdriver?

```
driver.maximize_window()
```

4. How to print title of the page and current url?

```
print(driver.title)
print(driver.current_url)
```

5. How to invoke Firefox and Edge browser?

```
# Firefox
from selenium.webdriver.firefox.service import Service
service_obj = Service(("path\to\geckodriver.exe"
driver = webdriver.Firefox(service=service_obj)
```

```
# Edge
from selenium.webdriver.edge.service import Service
service_obj = Service("path\to\msedgedriver.exe")
driver = webdriver.Edge(service=service_obj)
```

# Section 10:

1. Which locator's strategies are supported by Selenium?
   ID
   class name
   name

**Xpath**
**tagName**
**link text**
**Partial link text**
**CSS**

2. **How to define locators?**

```
driver.find_element(By.<locator strategies>, "//tagName[@attribute='value']")

Example,
driver.find_element(By.XPATH, "//input[@type='email']")
```

# Section 11:

1. **Websites for practice?**
   - https://rahulshettyacademy.com/angularpractice/
   - https://rahulshettyacademy.com/AutomationPractice/#/
   -https://rahulshettyacademy.com/seleniumPractise/#/

2. **How to handle static dropdown?**
   **-** Using Select class

```
dropdown = Select(driver.find_element(By.ID, "exampleFormControlSelect1"))
```

3. **What is the import statement for Select class?**

```
from selenium.webdriver.support.select import Select
```

4. **How to select option from the static dropdown using Select class?**
   **-** There are 3 methods which can be used to select required option from static dropdown

```
select_by_index(index)
select_by_value("value")
select_by_visible_text("visible text")
```

5. **Below is the example on how to select option from dropdown using select_by_index()**

```
from selenium.webdriver.support.select import Select

dropdown = Select(driver.find_element(By.ID, "exampleFormControlSelect1"))
dropdown.select_by_index(1)
```

6. **How to select a particular option using for loop?**

```
#driver.find_elements - This will get the list of web elements(checkboxes)
```

```
checkboxes = driver.find_elements(By.XPATH, "//input[@type='checkbox']")

#Iterating checkboxes using for loop and checking the checkbox 'option3'
for checkbox in checkboxes:
        if checkbox.get_attribute("value") == "option3":
                checkbox.click()
```

7. **How to select a radio button using index?**

```
radiobuttons = driver.find_elements(By.XPATH, "//input[@class='radioButton']")
radiobuttons[2].click()
```

8. **Which method is used to verify whether the checkbox/radio button is selected?**

```
is_selected()       #It will return Boolean - true or false
```

9. **How to handle java/javascript alert?**

```
driver.switch_to.alert
```

10. **How to extract text from alert?**

```
#Switch to alert
alert = driver.switch_to.alert

#Extract text from alert
alertMessage = alert.text

#Print alert text
print(alertMessage)
```

# Section 12:

1. **How to extract text from alert?**
   - Waits helps to facilitate synchronization between script actions and dynamic web elements
   - There are 3 wait mechanism - Implicit Wait, Explicit Wait, and Fluent Wait

2. **What is implicit waits and how to define it?**
   - Implicit Wait in Selenium is a global wait that applies to all elements in the script

```
driver.implicitly_wait(seconds)
```

3. **What is explicit wait and how to define it?**
   - Explicit Wait in Selenium is a more granular approach that allows you to wait for a specific condition or element.
   - Explicit waits are applied to individual elements and provide better control and precision.

```
#WebDriverWait is class
#It takes 2 arguments - web driver and seconds(time to wait)
wait = WebDriverWait(driver, 10)

#Different Expected Conditions - In this case we have used -
presence_of_element_located
wait.until(expected_conditions.presence_of_element_located((By.XPATH,
"//a[@class='promoInfo']")))
```

# Section 13:

**Assignment 2:**

```
from time import sleep

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
```

```
expectedVegetables = ['Cucumber - 1 Kg','Raspberry - 1/4 Kg','Strawberry - 1/4
Kg']
actualVegetables = []
service_obj = Service("path\to\chromedriver.exe")
driver = webdriver.Chrome(service=service_obj)

driver.maximize_window()
driver.implicitly_wait(2)
driver.get("https://rahulshettyacademy.com/seleniumPractise/#/")

driver.find_element(By.XPATH, "//input[@type='search']").send_keys("ber")
sleep(3)
results = driver.find_elements(By.XPATH, "//div[@class='products']/div")
count = len(results)
print(count)
assert count > 0

#Assignment 1:
for result in results:
        actualVegetables.append(result.find_element(By.XPATH, "h4").text)
        result.find_element(By.XPATH, "div/button").click()
print(actualVegetables)
assert expectedVegetables == actualVegetables
driver.find_element(By.XPATH, "//img[@alt='Cart']").click()
driver.find_element(By.XPATH, "//button[text()='PROCEED TO
CHECKOUT']").click()


#Sum Validation
prices = driver.find_elements(By.XPATH, "//td[5]//p[@class='amount']")
sum = 0
for price in prices:
        sum = sum + int(price.text)
print(sum)
totalAmount = int(driver.find_element(By.XPATH,
"//span[@class='totAmt']").text)
assert sum == totalAmount


driver.find_element(By.XPATH,
"//input[@class='promoCode']").send_keys("rahulshettyacademy")
```

```
driver.find_element(By.XPATH, "//button[@class='promoBtn']").click()

#Explicit wait - It is used to wait explicitly based on the given time for a
particular element
wait = WebDriverWait(driver, 10)
wait.until(expected_conditions.presence_of_element_located((By.XPATH,
"//span[@class='promoInfo']")))
print(driver.find_element(By.XPATH, "//span[@class='promoInfo']").text)

#Assignment 2: The value is in decimal hence converted value to float
totalAfterDis = float(driver.find_element(By.XPATH,
"//span[@class='discountAmt']").text)
assert totalAfterDis < totalAmount
```

# Section 14, 15:

1. **How to mouse hover on an element?**

```
#ActionChain - To perform mouse actions
#perform() is always required to perform action
action = ActionChains(driver)

action.move_to_element(driver.find_element(By.XPATH,
"//button[@id='mousehover']")).perform()      #mouse hover
```

2. **How to right click on an element?**

```
#ActionChain - To perform mouse actions
#perform() is always required to perform action
action = ActionChains(driver)
```

```
action.context_click(driver.find_element(By.LINK_TEXT, "Top")).perform()
#Right Click
```

3. **How to perform mouse hover + click on an element?**

```
#ActionChain - To perform mouse actions
#perform() is always required to perform action
action = ActionChains(driver)

action.move_to_element(driver.find_element(By.LINK_TEXT,
"Reload")).click().perform()      #mouse hover + click
```

4. **What is the import statement for Actions class?**

```
from selenium.webdriver import ActionChains
```

5. **How to switch to windowHandles?**

```
#Get all window handles
windowsOpened = driver.window_handles

#Switched to next window or child window using index
driver.switch_to.window(windowsOpened[1])
```

6. **How to switch to frame?**

```
# Switch to frame using frame id or frame name
driver.switch_to.frame("id" or “frame name”)
```

7. **How to scroll at a certain axis on the web page using Javascript Executor?**

```
#JavascriptExecutor to scroll down at certain point of the page
driver.execute_script("window.scrollBy(X axis,Y axis);")

driver.execute_script("window.scrollBy(0,700);")
```

8. **How to scroll at the bottom of the web page using Javascript Executor?**

```
#JavascriptExecutor to scroll at the bottom of the page
driver.execute_script("window.scrollBy(0,document.body.scrollHeight);")
```

9. **Which method helps to sort a list?**

```
sort()
```

10. **What is ChromeOptions?**
    - ChromeOptions is used for customizing the ChromeDriver session

11. **How to run scripts on headless mode using ChromeOptions?**

```
#ChromeOptions invocation using werbdriver.ChromeOptions()
chrome_options = webdriver.ChromeOptions()

chrome_options.add_argument("--headless")
```

12. **How to ignore certificate error using ChromeOptions?**

```
#ChromeOptions invocation using werbdriver.ChromeOptions()
chrome_options = webdriver.ChromeOptions()

chrome_options.add_argument("--ignore-certificate-errors")
```

13. **How to start browser in maximized mode using ChromeOptions?**

```
#ChromeOptions invocation using werbdriver.ChromeOptions()
chrome_options = webdriver.ChromeOptions()

chrome_options.add_argument("--start-maximized")
```

## Section 16: Complete code is provided in the last lecture of this section.

## Section 17, 18 and 19:

1. **Which Python unit test framwork is taught is this course?**

- Pytest

2. **What is Pytest?**
   - The pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

3. **How to install pytest using pip?**
   - pip install pytest

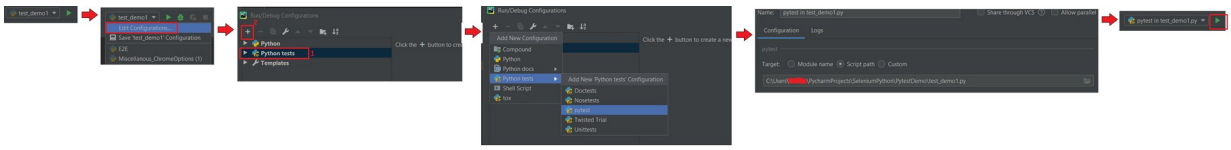
4. **How to check pytest version?**
   - pytest –version

5. What is pytest standard to be followed while working with pytest file?
   - In pytest, python file name should always start with test_ or end with _test
   - logic should be written within a function/method
   - function/method name should start with test_
   - class name should start with Test

6. Example – pytest file

```
#Method declaration
def test_firstProgram():
        print("Hello")
```

7. **How to execute a pytest file?**



8. **How to execute python file in pytest from command prompt?**
   - cd to Python package(where pytest file is present)
   - py.test -v –s

9. **What is –v and –s?**
   - -v: It provides more information about test case execution
   - -s: To display print statement on command prompt

10. **How to run a particular python file in pytest from command prompt?**
    - py.test python_file_name -v –s

11. **How to execute test cases using method/function name from command prompt?**
    - py.test -k <partial_method_name> -v –s

12. **Tags in pytest**

```
#@pytest.mark.<tagName>: this will help to group methods/functions
@pytest.mark.smoke
def test_greet():
        print("Good morning!")
```

13. **How to execute grouped methods/functions from command prompt?**
    **-** py.test -m <pytest_mark_name> -v –s

14. **How to skip a method/function?**

```
@pytest.mark.skip
def test_CreditCard1():
        print("Credit Card 1")
```

15. **How to execute a method/function without reporting it in test report?**

```
@pytest.mark.xfail
def test_tag():
        print("Tag 1")
```

16. **What are fixtures in pytest?**
    **-** In testing, a fixture provides a defined, reliable and consistent context for the tests. This could include environment (for example a database configured with known parameters) or content (such as a dataset)

17. **How to declare a fixture?**
    **-** @pytest.fixture

18. **Example:**

```
#fixture decalred on setup method/function
@pytest.fixture()
def setup():
        print("Setup")

#Fixture method 'setup' is passed as an argument to this method
#Before executing this method, it will execute setup method
def test_firstFixtureDemo(setup):

        print("First demo")
```

19. **Which keyword helps to execute the set of code after the test case execution completion?**

- yeild

20. **What is conftest in pytest?**
    -The conftest.py file serves as a means of providing fixtures for an entire directory. Fixtures defined in a conftest.py can be used by any test in that package without needing to import them (pytest will automatically discover them).

21. **Data driven fixtures to load test data into tests**

*Conftest.py*

```
#Declared fixture to dataLoad method
@pytest.fixture()
def dataLoad():
        print("user profile data is being created")
        #Return set of values
        return ["Rahul", "Shetty", "QA Click"]
```

*test_demo.py*

```
#It will first execute dataLoad fixture from conftest file
@pytest.mark.usefixtures("dataLoad")
class TestExample:
        #dataLoad: It has values returned from conftest dataLoad fixture
        def test_fixturesdemo(self, dataLoad):
                print(dataLoad[0])    #It will print Rahul
```

22. **How to install pytest-html package?**
    - pip install pytest-html

23. **How to execute test cases and generate pytest-html report?**
    - Navigate to the folder where pytest files are present
    - Hit the command: *pytest --html=report.html*

24. **How to create a log file for logging messages?**

```
# FileHandler is used to specify file path. This will create Logs.log in
pytestDemo package
filehandler = logging.FileHandler('Logs.log')
```

```
# Formatter will help to format the log, asctime: timestamp, levelname: debug,
info, error etc., name: File name, messsage: message
```

```
formatter = logging.Formatter("%(asctime)s :%(levelname)s :%(name)s :%
(message)s")
```

```
# filehandler object will have the format
filehandler.setFormatter(formatter)
```

```
# created object of logging and __name__    will help use to get the file name
logger = logging.getLogger(__name__)
```

```
# filehandler is passed in logger which has file path and file format
logger.addHandler(filehandler)
```

```
# setLevel will print the value from Info. It will skip debug
logger.setLevel(logging.INFO)
```

```
logger.warning("Warning message")
```