

Assignments for Programming in Python



Author(s)	Sahana Kumaraswamy
Authorized by	Pramod Panda
Creation/Revision Date	Dec 2015
Version	1.0

COPYRIGHT NOTICE

© 2016 Infosys Limited, Bangalore, India. All Rights Reserved.

Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.

Education, Training and Assessment Department
Infosys Limited
Electronics City
Hosur Road
Bangalore – 561 229, India.

Tel: 91 80 852 0261-270

Fax: 91 80 852 0362

www.infosys.com

<mailto:ETA@infosys.com>

Document Revision History

Version	Date	Author(s)	Reviewer(s)	Description
1.0	Dec 2015	Sahana Kumaraswamy	Dr. P. Suresh Kalpana Balaraman	Initial Draft

CONTENTS

COPYRIGHT NOTICE.....	i
Document Revision History	ii
CONTENTS	iii
Assignments for Programming in Python.....	1
Context.....	1
Guidelines.....	1
Programming Fundamentals in Python - Assignments.....	1
Assignment 1: Using Eclipse IDE to create and execute Python Program - Guided Activity	1
Assignment 2: Observations from a real world problem - Guided Activity	10
Assignment 3: Programming constructs in Python - Guided Activity.....	11
Assignment 4: Programming constructs in Python - Quiz.....	11
Assignment 5: Programming constructs in Python - Demo.....	12
Assignment 6: Programming constructs in Python – Hands on practice	13
Assignment 7: Programming constructs in Python - Hands - on - Practice	13
Assignment 8: Programming constructs in Python - Guided Activity	14
Assignment 9: id() and type() functions - Quiz	14
Assignment 10: Coding Standards - Guided Activity.....	16
Assignment 11: Control Structures – Observations from a real world problem - Guided Activity	16
Assignment 12: Control Structures - Demo.....	17
Assignment 13: Control Structures - Guided Activity.....	17
Assignment 14: Control Structures - Guided Activity	18
Assignment 15: Control Structures – Hands – on – Practice	20
Assignment 16: Control Structures – Hands - on - Practice	20
Assignment 17: Iteration Control Structures - Guided Activity	21
Assignment 18: break statement - Demo	23
Assignment 19: continue statement - Demo	24
Assignment 20: Iteration Control Structure – Debugging - Guided Activity	25
Strings - Assignments	25
Assignment 21: Strings – Observations from Retail Application - Guided Activity	25
Assignment 22: Strings – Demo	26
Assignment 23: Strings - Quiz.....	27
Assignment 24: Strings – Hands - on - Practice	28
Assignment 25: Strings – Hands – on - Practice.....	28
Assignment 26: Strings – Hands – on - Practice.....	29
Assignment 27: Strings – Hands - on - Practice	29
Assignment 28: Operations on Tuples - Demo.....	30

Assignment 29: Tuples – Hands – on - Practice	31
Lists - Assignments.....	32
Assignment 30: Accessing Elements from Lists - Demo.....	32
Assignment 31: Lists – Hands - on - Practice	33
Assignment 32: Lists – Hands - on - Practice	33
Assignment 33: Lists - Hands - on - Practice.....	34
Assignment 34: Sets - Demo.....	35
Assignment 35: Sets – Hands - on - Practice	37
Assignment 36: Dictionary - Demo	37
Assignment 37: Dictionary – Hands - on - Practice.....	38
Assignment 38: Functions – Pass by Reference - Demo	39
Assignment 39: Functions – Pass by Reference – Hands - on - practice	40
Assignment 40a: Recursion – Demo.....	42
Assignment 40b: Recursion – Hands – on - Practice	43
Assignment 41: Recursion – Hands – on - Practice	44
Assignment 42: Recursion – Hands – on - Practice	45
Assignment 43: Strings built - in functions - Guided Activity	45
Assignment 44: Lists built-in functions - Guided Activity	46
Assignment 45: Lists – Stack – Observation from real world -Guided Activity	47
Assignment 46: Lists – Stack - Demo.....	48
Assignment 47: Lists – Stack - Hands - on - Practice	52
Assignment 48: Lists – Queue - Guided Activity	52
Assignment 49: Regular Expression – match() - Guided Activity.....	56
Assignment 50: Regular Expression – findall() - Guided Activity	58
Assignment 51: Regular Expression – search() and replace() - Guided Activity	59
Assignment 52: Regular Expression - Hands - on - Practice	61

Assignments for Programming in Python

Context

This document contains assignments to be completed as part of the hands on session for the course - Programming in Python

Guidelines

- The assignments guide has been designed to give hands on experience to map/apply the concepts learnt in the theory session with practical assignments.
- The assignments have been categorized into solved assignments to hand hold a beginner, partially solved assignments to begin trying out on their own and unsolved assignments to let learners write code completely on their own
- These assignments contain coding exercises, debugging exercises, coding standards exercises assignments
- The case study based assignments are threaded, which can be built incrementally. This will help understanding the concepts and building a complete application
- The estimated time would help a learner to solve problems given a deadline
- The assignments need to be completed as instructed by the facilitators
- **Assignments marked as Demo can be used by the faculty during lecture and students need to compile and execute it using Eclipse**

Programming Fundamentals in Python - Assignments

Assignment 1: Using Eclipse IDE to create and execute Python Program - Guided Activity

Objective: Learn how to use the Eclipse IDE to create and execute Python programs.

About Eclipse

Eclipse is a multi-language Integrated Development Environment (IDE) and it has an extensible plug-in system. It can be used to develop applications in Python and also other programming languages including C, C++, Ruby, Perl, Python, COBOL, PHP, Java etc. by means of various plug-ins.

Using the Eclipse IDE

Starting Eclipse and Giving Workspace Information

- To start the Eclipse IDE, double click the Eclipse.exe, You may create a shortcut on your desktop

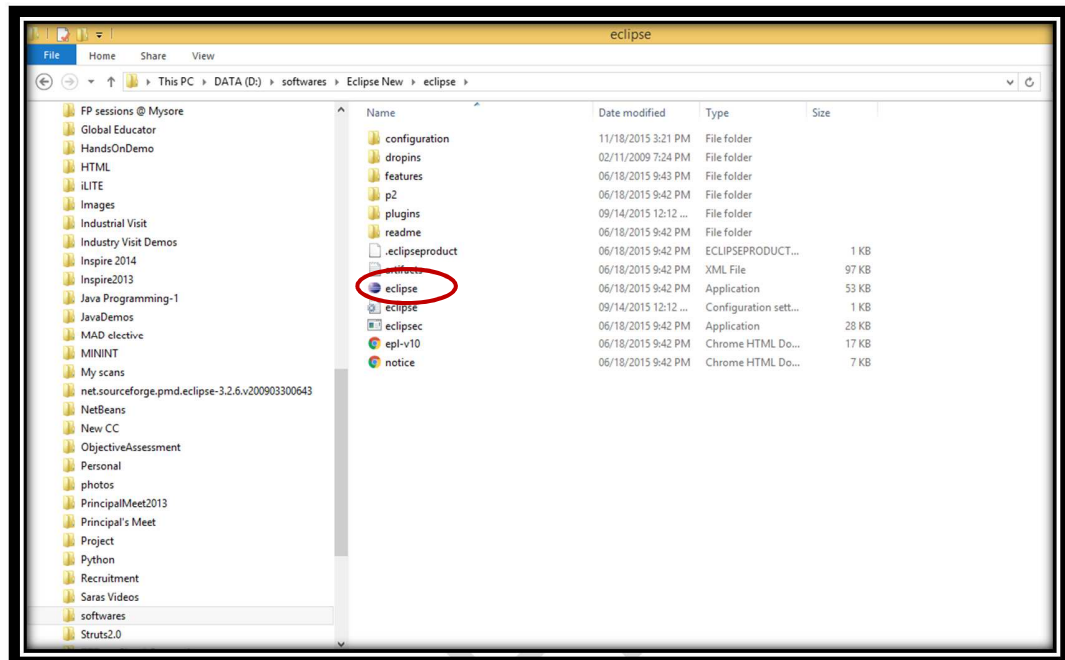


Fig. 1 Starting Eclipse



Fig. 2 Eclipse Loading

Once started a prompt will open which will ask you to enter the workspace information as shown in Fig. 3. A workspace is a folder where all the projects created in Eclipse are saved. Create a folder in your desired path and give the correct path as shown below.

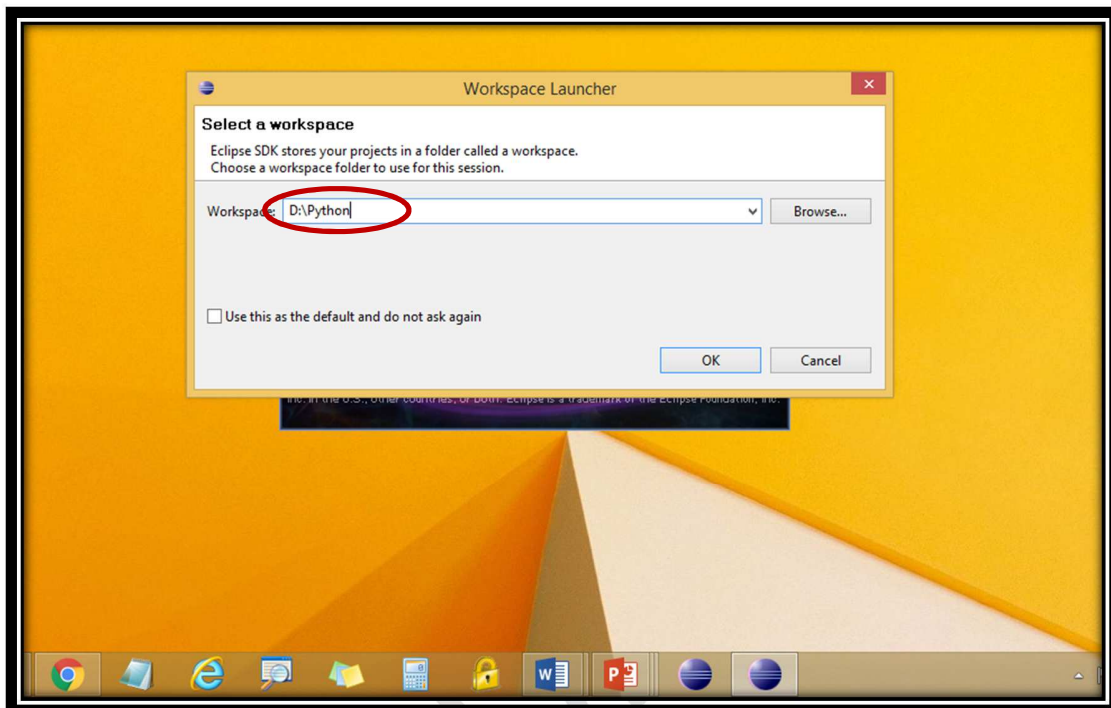


Fig. 3 Work Space

Once loaded, you will see the Welcome tab as shown in Fig. 4. Close this Welcome tab to go to the Project Window.

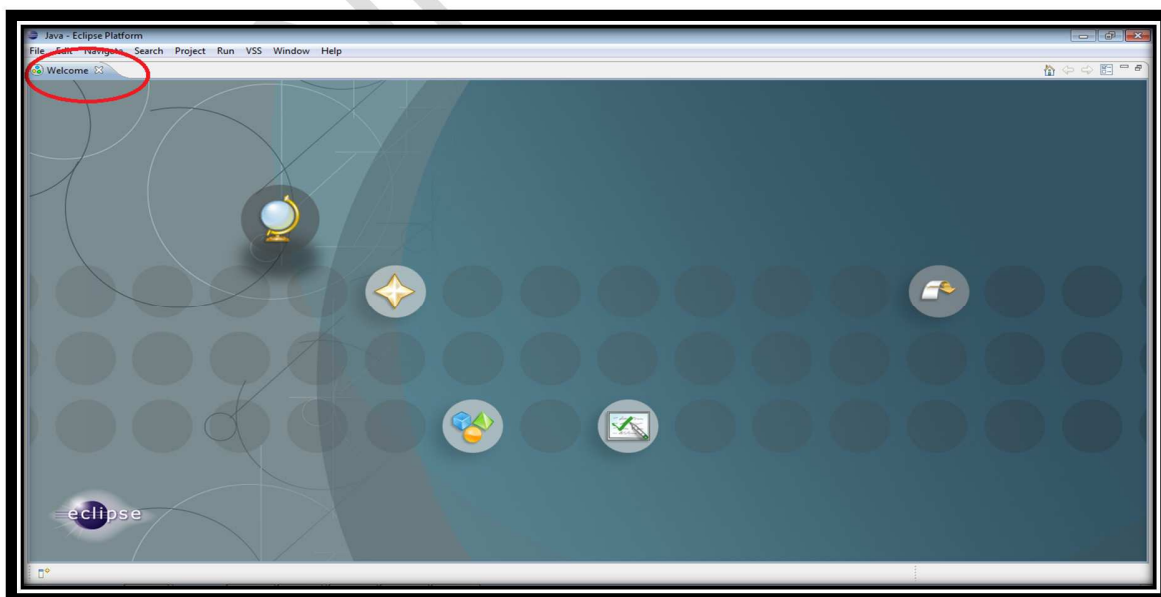


Fig. 4 Welcome Screen

Next go to the Window menu and select the perspective as Python perspective as shown in Fig. 5.1 and Fig 5.2. (Window → Open Perspective → PyDev)

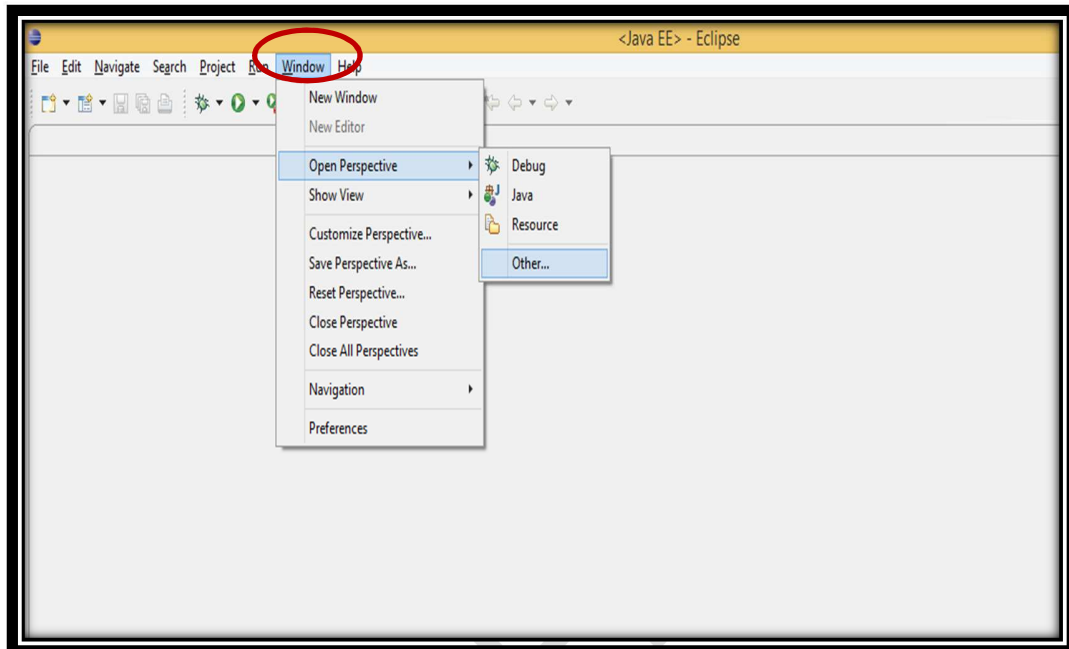


Fig. 5.1 Changing Perspective

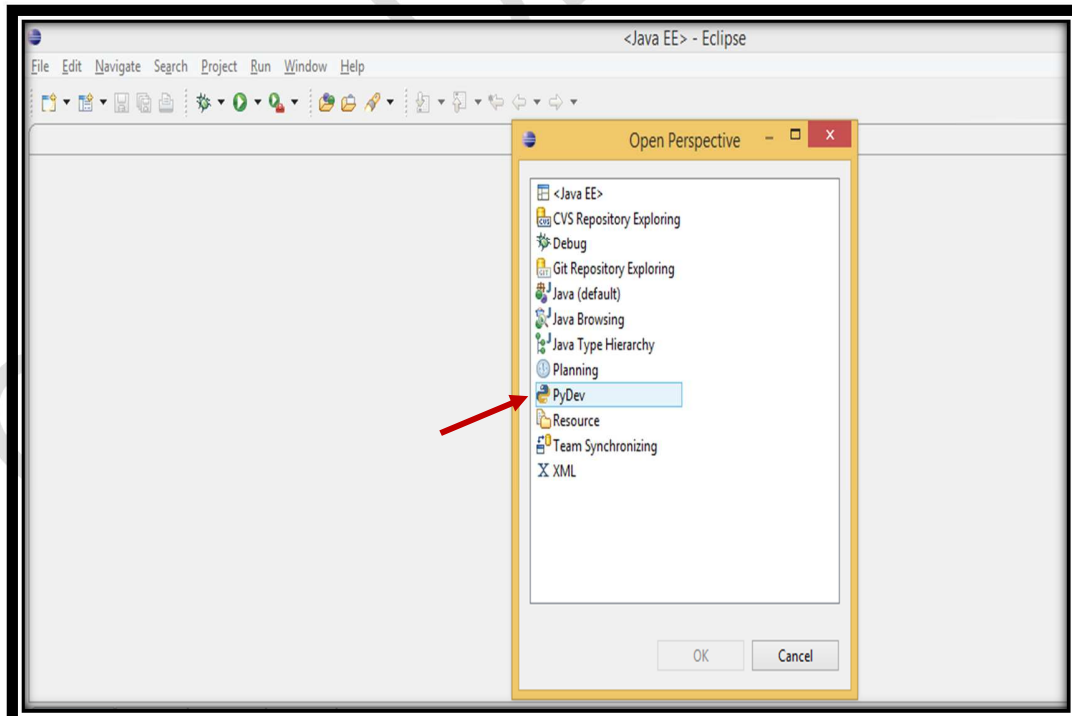


Fig. 5.2 Changing Perspective

Selecting the PyDev perspective will give you the screen as shown in Fig. 6.

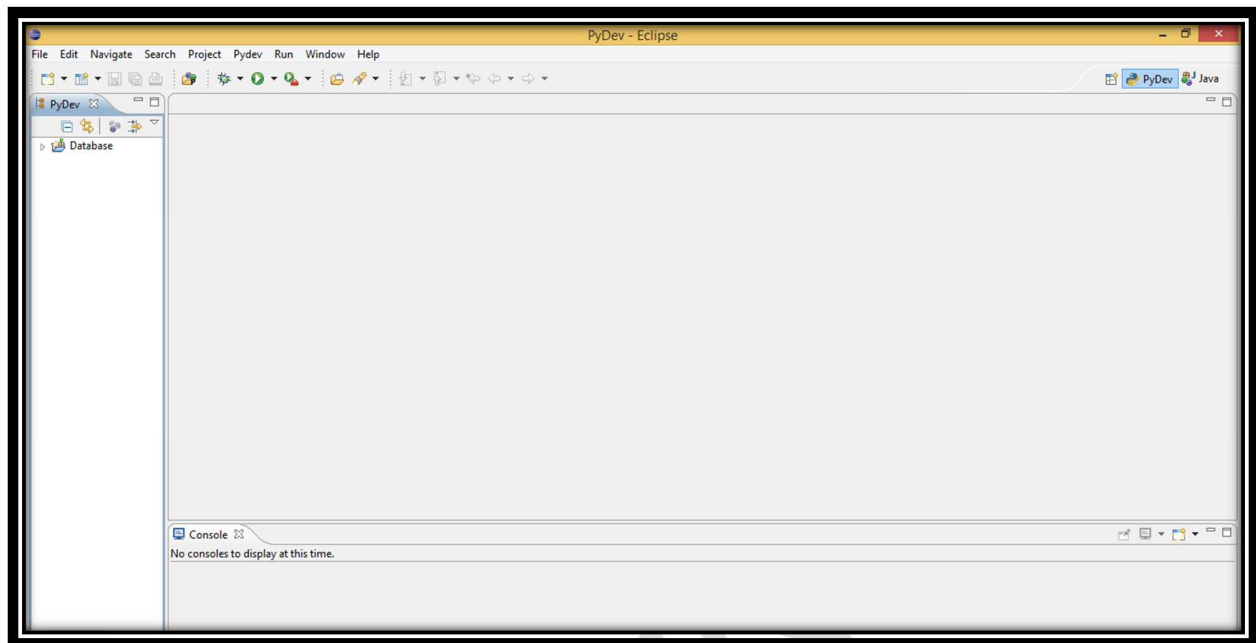


Fig. 6 Project Explorer

Creating a New Project

- In Eclipse a workspace can contain one or more projects and each project usually contains a Python application. Each application may contain one or more Python files.
- To create a new project, go to File → New → Project as shown in Fig. 7.

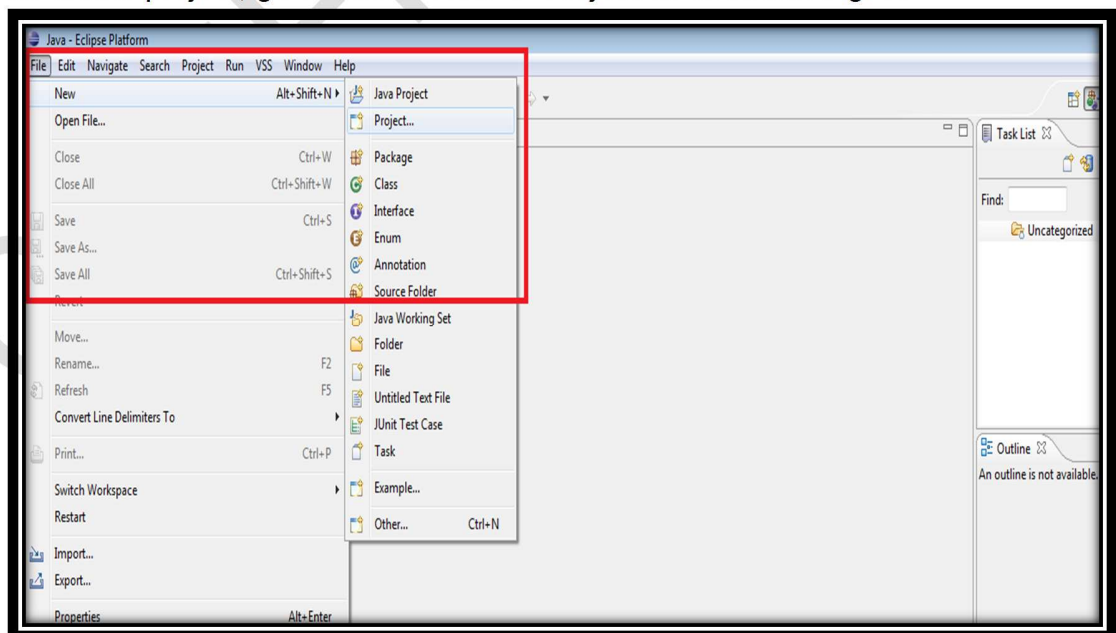


Fig. 7 New Project

- Select the option PyDev Project under the PyDev category as shown in Fig. 8.

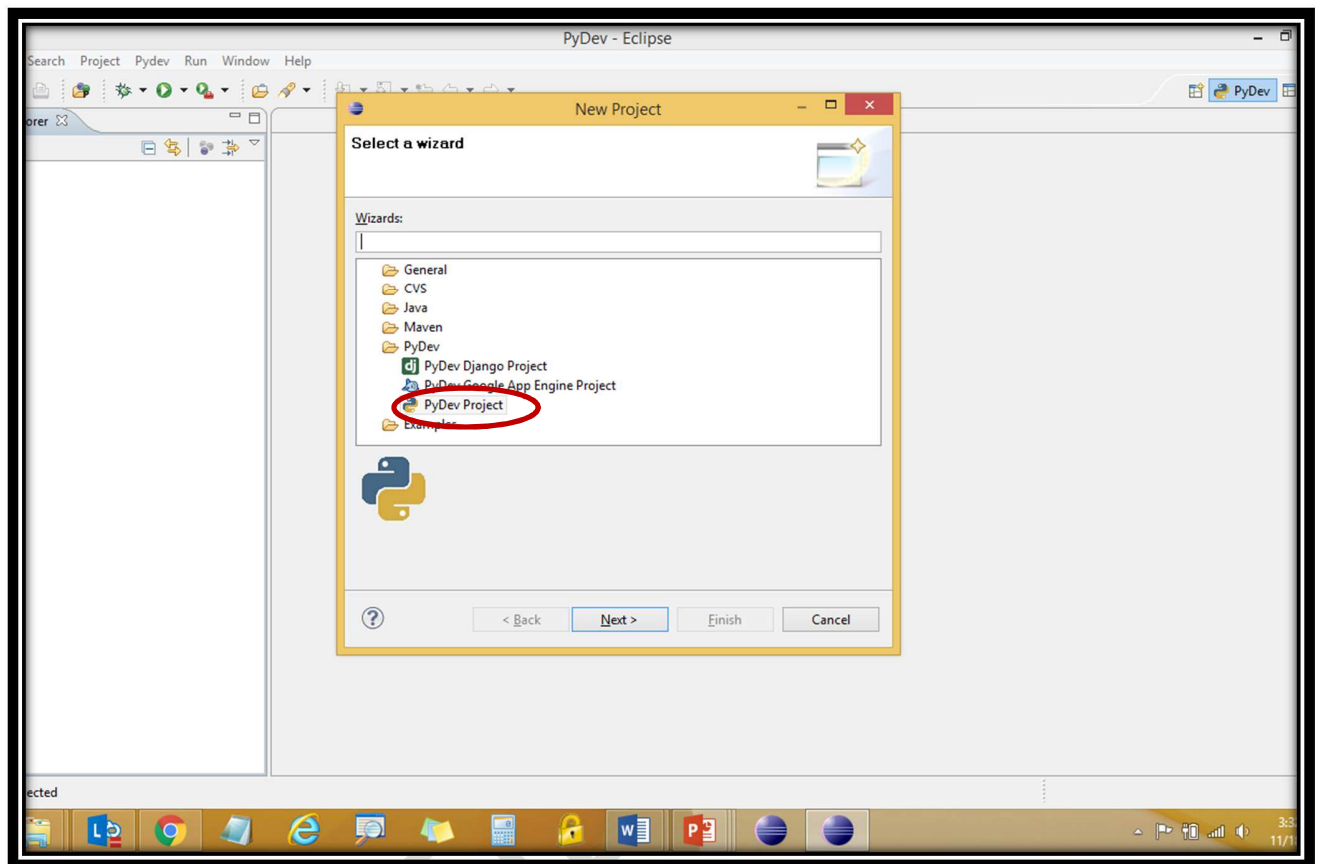
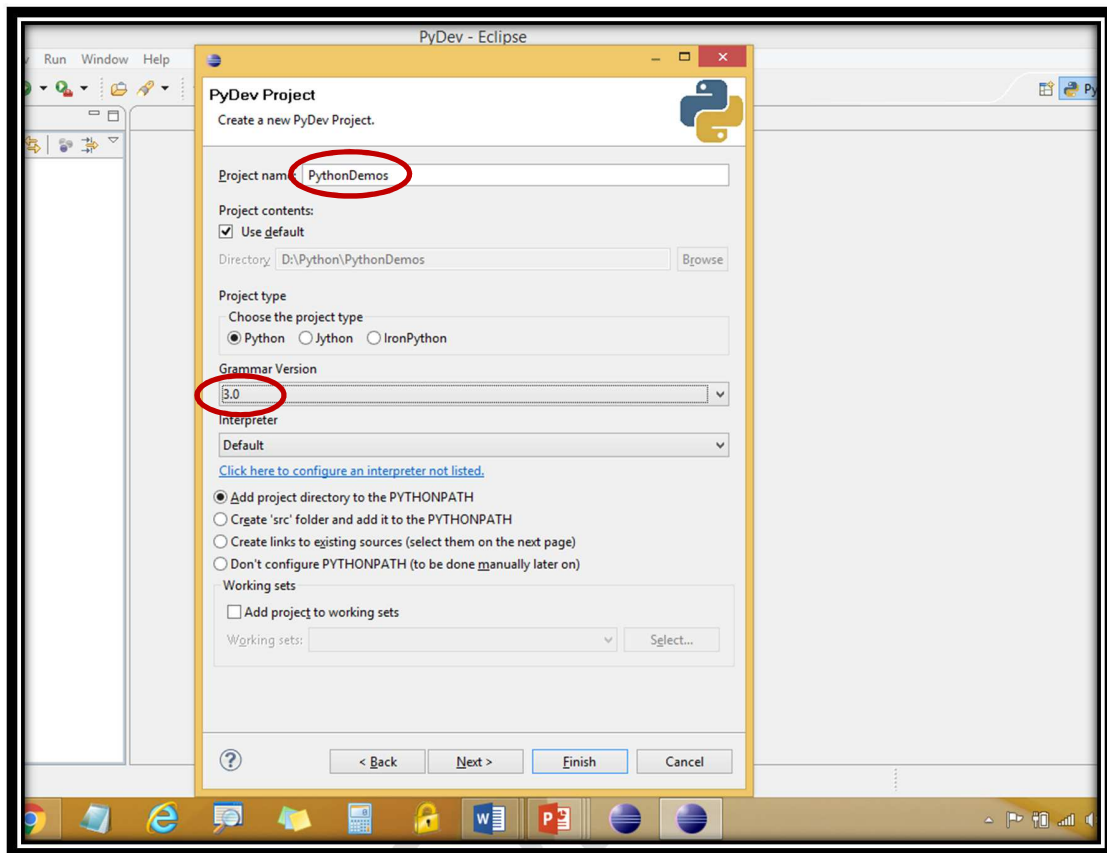
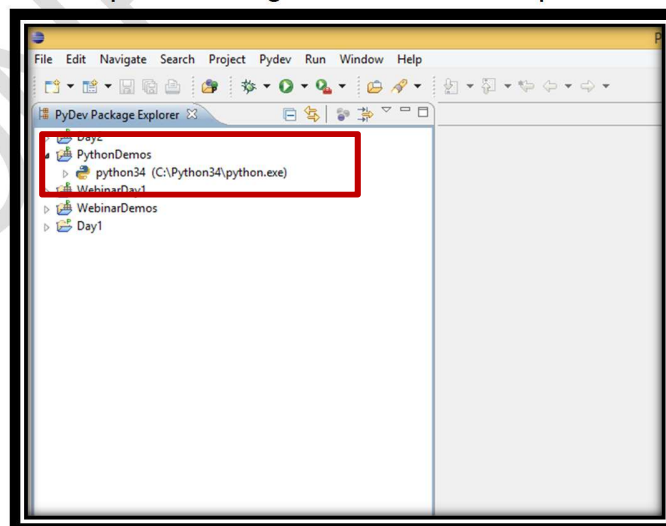


Fig. 8 Project Type – PvDev

- Enter a relevant project name, Set Grammer Version to 3.0
- Select the interpreter from drop down menu if listed or click on “*click here to configure an interpreter not listed*”
- You may configure the interpreter through *quick auto config option* or *manual config* by manually entering the path of the Python interpreter
- After configuring interpreter, click on finish button as shown in Fig. 9.

**Fig. 9 Project Details**

- This will create a project and you can see the same in the package explorer along with the path of the interpreter configured in the last step as shown in Fig. 10.

**Fig. 10 Project Directory Structure**

Writing Your First Python Program

- To write your first Python program you must first create a PyDev Module. Right click on the project and go to New → PyDev Module as shown in Fig. 11.

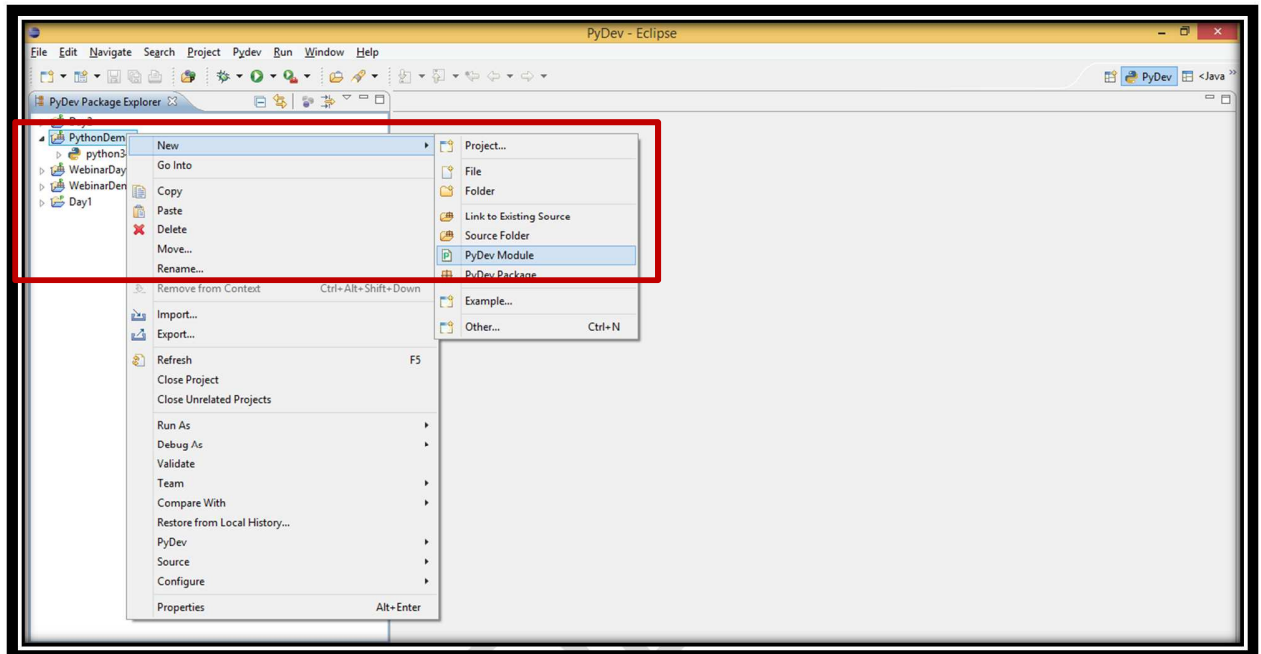


Fig. 11 New PyDev Module

- Enter the Package name and PyDev Module name as shown in Fig. 12 and then click on finish. Packages are like directories and PyDev modules are like files. Packages will be explained later.

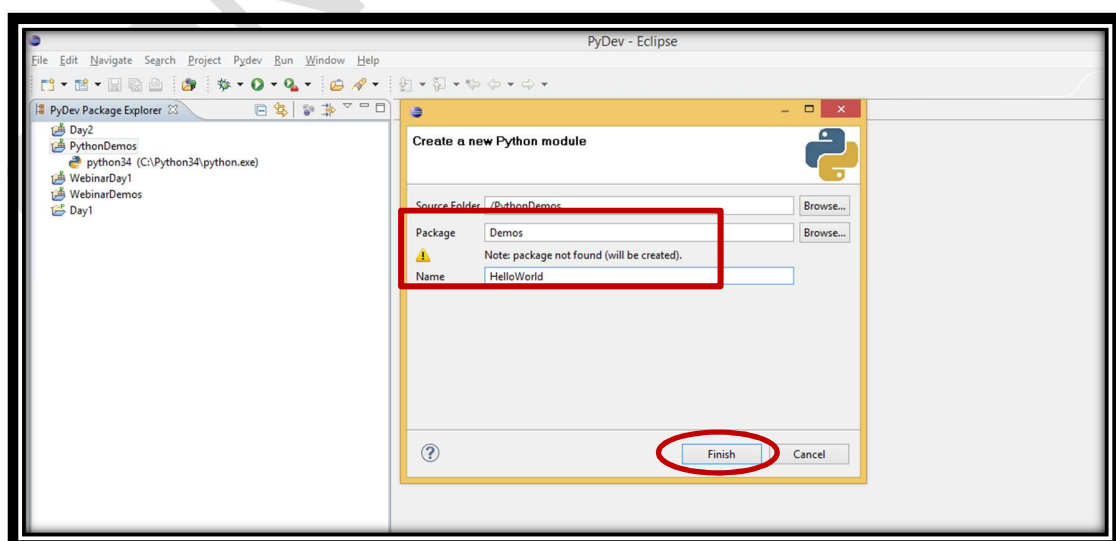


Fig. 12 Enter Package and Module Details

- This will add a Package and a PyDev module in the package to the project as shown in Fig. 13.
- You can now write your Python program as shown in Fig.13
- You can also create new PyDev modules in the package created

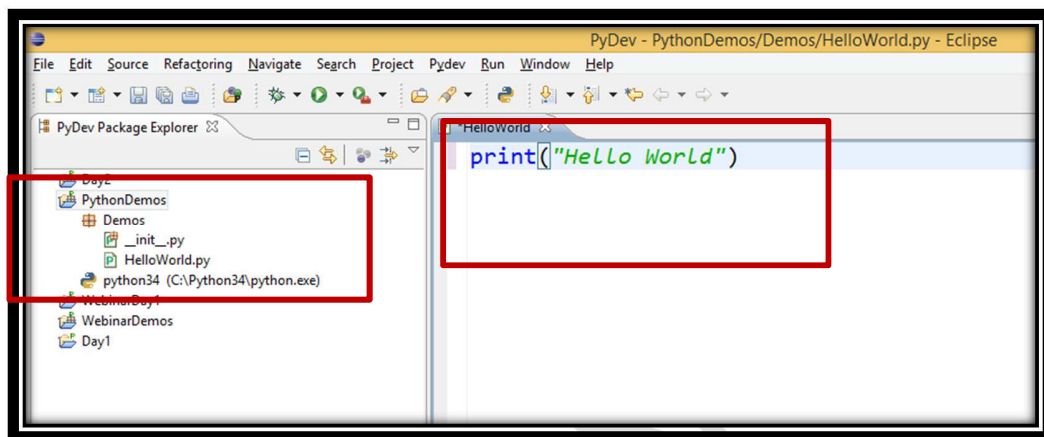


Fig. 13 Sample Program

Executing Your Python Program

- To execute a PyDev Module, right click on the module-name in the project explorer and go to Run As → Python Run as shown in Fig. 14.

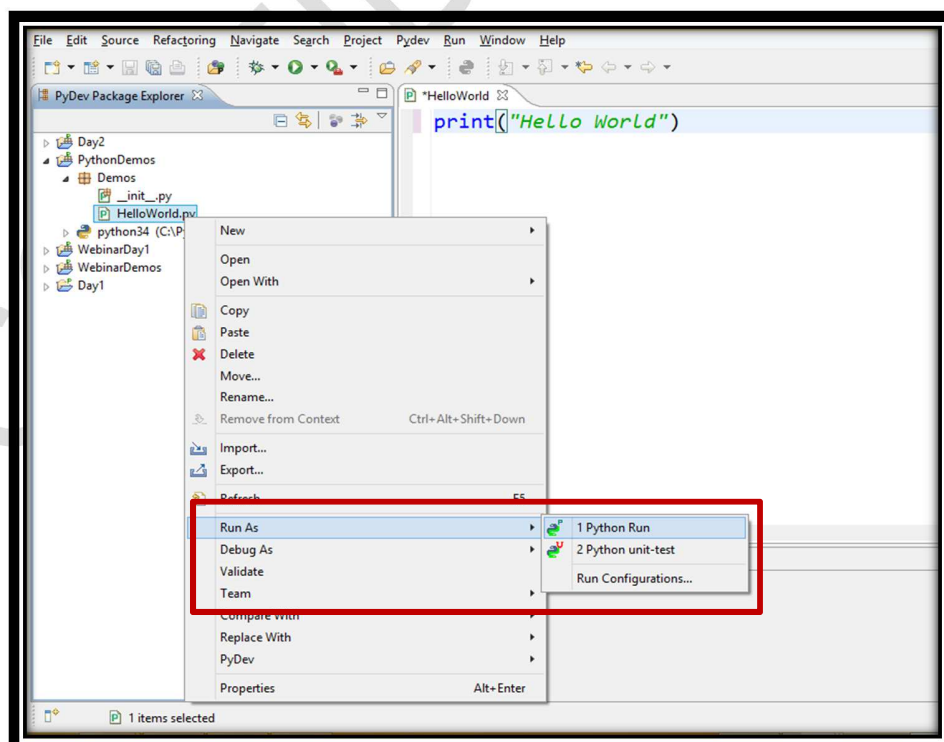


Fig. 14 Executing the Program

- The output of the program will be displayed in the Console Tab as shown in Fig.15.

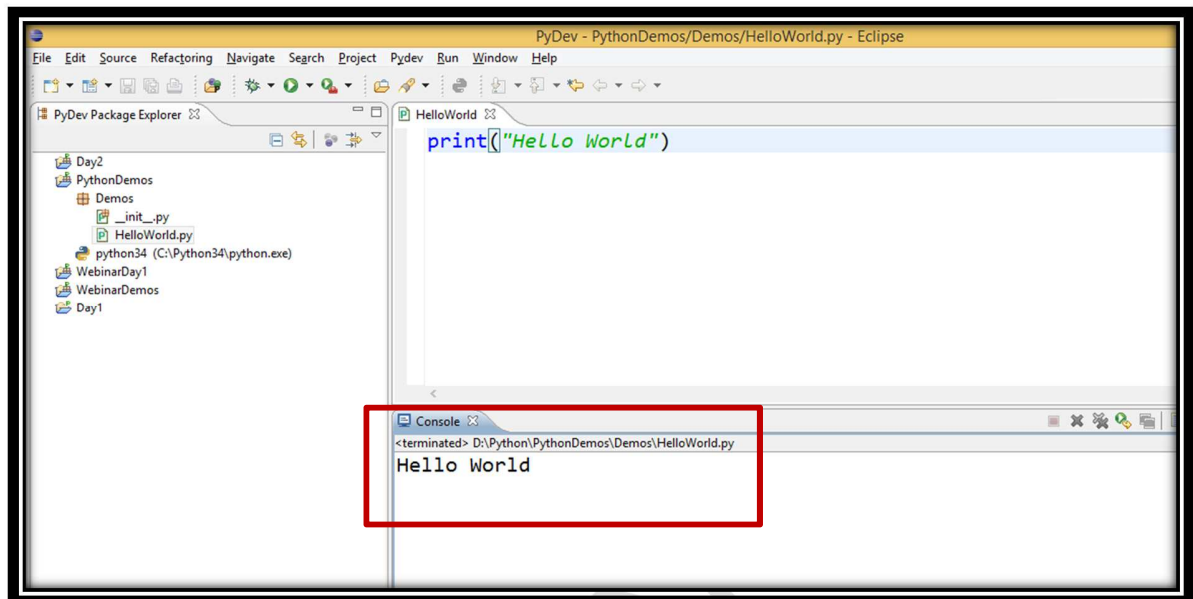


Fig. 15 Output Window

Estimated time: 15 min

Summary of this assignment: You have now learnt how to create projects, how to create and execute a simple Python program using Eclipse IDE.

Assignment 2: Observations from a real world problem - Guided Activity

Objective: Given a real world problem, be able to understand the need for programming fundamentals such as identifiers, variables, data types etc

Problem Description: A retail store management wants to automate the process of generating the bill amount for its customers. As an initial step, they want to initialize the bill details of a customer as given below:

Bill id should be 1001, customer id should be 101 and bill amount should be 199.99.

After initializing, all the values must be displayed in the format given below:

bill_id: 1001

customer_id: 101

bill_amount: Rs.199.99

Analyze the above problem statement and answer the following questions:

1. What do you think is needed to write a program to implement the solution for the above problem statement?

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have understood the need of a high level programming language and programming fundamentals such as identifiers, variables, data types, operators etc

Assignment 3: Programming constructs in Python - Guided Activity

Objective: Given a real world problem, be able to identify the data types of the variables required to solve it

Problem Description: For the previous assignment, identify the data types that may be used to represent bill id, customer id and bill amount.

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have learnt to identify the data type for the variables based on the real world problem

Assignment 4: Programming constructs in Python - Quiz

Objective: Given a real world problem, be able to identify the variables and operators required to solve the problem and implement it using a high level programming language like Python

Problem Description: Predict the output of following Python Statements.

```
print ("Value of e is %0.1f" , 2.713 )
print ("Value of e is %0.1f" %2.713 )
print ("Programming in Python: Version %d" %3.5 )
print ("%20s : %d" % ("Python 3.0 is also known as Python",3000.57 ))
x=2
print ("{0:2d} {1:3d} {2:4d}".format(x, x*x, x*x*x))
```

Output:

```
Value of e is %0.1f 2.713
Value of e is 2.7
Programming in Python: Version 3
Python 3.0 is also known as Python 3000
 2   4   8
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt to make use of format specifiers for different type of data variables based on the real world problem

Assignment 5: Programming constructs in Python - Demo

Objective: Given a real world problem, be able to identify the variables and operators required to solve the problem and implement it using a high level programming language like Python

Problem Description: Write a Pseudocode and Python program to implement the real world problem discussed in Programming constructs in Python Assignment 2. Compile and execute the program using Eclipse IDE.

You may want to refer to Programming constructs in Python Assignment 1 to understand how Eclipse IDE may be used for writing and executing a Python program.

INITIALIZE_VARIABLES_DISPLAY

1. `bill_id = 1001`
2. `customer_id = 101`
3. `bill_amount = 199.99`
4. **display** "Bill Id:", `bill_id`
5. **display** "Customer Id:", `customer_id`
6. **display** "Bill Amount:Rs.", `bill_amount`

```
bill_id = 1001
customer_id = 101
bill_amount = 199.99
print("Bill Id:%d" %bill_id)
print("Customer Id:%d" %customer_id)
print("Bill Amount:Rs.%f" %bill_amount)
```

Output:

```
Bill Id: 1001
Customer Id: 101
Bill Amount:Rs. 199.990000
```

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have understood how to implement the solution for a simple real world problem using variables and operators

Assignment 6: Programming constructs in Python – Hands on practice

Objective: Given a real world problem, be able to identify the variables and operators required to solve the problem and implement it using a high level programming language like Python

Problem Description: The finance department of a company wants to calculate the monthly pay of one of its employee. Monthly pay should be calculated as mentioned in the below formula and display all the employee details.

Monthly Pay = Number of hours worked in a week * Pay rate per hour * No. of weeks in a month

Note:

The number of hours worked by the employee in a week should be considered as 40,
Pay rate per hour should be considered as Rs.400 and
Number of weeks in a month should be considered as 4

Write Pseudo code and Python program in Eclipse to implement the above real world problem.

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt to implement the solution for a simple real world problem using variables and operators

Assignment 7: Programming constructs in Python - Hands - on - Practice

Objective: Given a real world problem, be able to identify the variables and operators required to solve the problem and implement it using a high level programming language like Python

Problem Description: Write the assignment statement to swap 2 numbers x and y? (Without using temp variable)

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have learnt to implement the solution for a simple real world problem using variables and operators

Assignment 8: Programming constructs in Python - Guided Activity

Objective: Given a real world problem, be able to identify the variables and operators required to solve the problem and implement it using a high level programming language like Python

Problem Description: Predict the output of following code snippet

```
num = 16
num1 = num/6
num2 = num//6
num3 = num//6.0
print(num1)
print(num2);
print(num3)
```

Output:

```
2.6666666666666665
2
2.0
```

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have learnt to implement the solution for a simple real world problem using variables and operators

Assignment 9: id() and type() functions - Quiz

Objective: Revisit the concept and usage of id() and type() functions

Problem Description:

Dry run the below code snippets and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Code – 1:

```
obj_x = 10
obj_y = obj_x

if ( id(obj_x) == id(obj_y) ):
    print("Address of obj_x and obj_y is same")
else:
    print("Address of obj_x and obj_y is not same")
```

Output:

```
Address of obj_x and obj_y is same
```

Code – 2:

```
obj_x = 10
obj_y = obj_x

if ( obj_x is obj_y ):
    print("obj_x and obj_y have same identity")
else:
    print("obj_x and obj_y do not have same identity")
```

Output:

```
obj_x and obj_y have same identity
```

Code – 3:

```
obj_x = 10
obj_y = obj_x
obj_y = obj_x + 1
if ( obj_x is not obj_y ):
    print("obj_x and obj_y do not have same identity")
else:
    print("obj_x and obj_y have same identity")
```

Output:

```
obj_x and obj_y do not have same identity
```

Code – 4:

```
int_a = 10
raw_input = input("Enter a number")
print("Type of int_a:", type(int_a))
print("Type of raw_input:", type(raw_input))
print(int_a + raw_input)
```

Output:

```
Enter a number 6
Type of int_a: <class 'int'>
Type of raw_input: <class 'str'>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have understood the concept and usage of `id()` and `type()` built-in functions.

Assignment 10: Coding Standards - Guided Activity

Objective: Given a set of source code and an identified set of best practices, be able to implement the techniques during coding

Problem Description: Coding standards are very important for maintenance of code and for understanding of the code. Identify the sections of the given program where the coding standards are not followed and correct them.

```
itemNo=1005
unitprice = 250
quantity = 2
amount=quantity*unitprice
print("Item No:",itemNo)
print("Bill Amount:",amount)
```

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have learnt Python coding standards

Assignment 11: Control Structures – Observations from a real world problem - Guided Activity

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: The scenario discussed in Programming Fundamentals Assignment 5 is revisited here. Suppose the retail store management now wants to provide 2% discount for all bill amounts above Rs.500 and for all other bill amount, a discount of 1%.

What do you think is needed to implement this scenario?

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have understood the need of operators and control structures using a real world problem

Assignment 12: Control Structures - Demo

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: Suppose the retail store management now wants to provide 2% discount for all bill amounts above Rs.500 and for all other bill amount, a discount of 1%. Write a Python program to implement the same?

```
bill_id = 1001
customer_id = 101
bill_amount = 200.0
discounted_bill_amount = 0.0
print("Bill Id:%d" %bill_id)
print("Customer Id:%d" %customer_id)
print("Bill Amount:Rs.%f" %bill_amount)
if bill_amount > 500:
    discounted_bill_amount = bill_amount - bill_amount * 2 / 100
else:
    discounted_bill_amount = bill_amount - bill_amount * 1 / 100
print("Discounted Bill Amount:Rs.%f" %discounted_bill_amount)
```

Note the use if else statement

Output:

```
Bill Id:1001
Customer Id:101
Bill Amount:Rs.200.000000
Discounted Bill Amount:Rs.198.000000
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have understood the implementation of operators and control structures using a real world problem

Assignment 13: Control Structures - Guided Activity

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

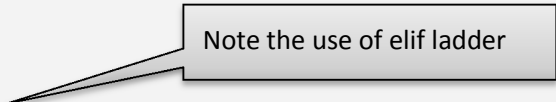
Problem Description: Suppose the retail store management now wants to provide discount for all bill amounts as mentioned below.

Assume bill amount will be always greater than 0.

Bill Amount	Discount %
>=1000	5
>=500 and <1000	2
>0 and <500	1

Write a Python program to implement the same?

```
bill_id = 1001
customer_id = 101
bill_amount = 1200.0
discounted_bill_amount = 0.0
discount = 0
print("Bill Id: %d" %bill_id)
print("Customer Id: %d" %customer_id)
print("Bill Amount:Rs. %f" %bill_amount)
if bill_amount >= 1000:
    discount = 5
elif bill_amount >= 500:
    discount = 2
else:
    discount = 1
discounted_bill_amount = bill_amount - bill_amount * discount / 100
print("Discounted Bill Amount:Rs. %f" %discounted_bill_amount)
```



Output:

```
Bill Id: 1001
Customer Id: 101
Bill Amount:Rs. 1200.000000
Discounted Bill Amount:Rs. 1140.000000
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have understood the implementation of operators and control structures using a real world problem

Assignment 14: Control Structures - Guided Activity

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: Suppose the retail store management now wants to provide discount for all bill amounts as mentioned below.

Customers can be considered to be valid, if their customer id is between 101 and 1000(both inclusive).

For valid customers, discount must be provided as per the table given below:

Bill Amount	Discount %
>=500	10

Assume for all other cases, discount is 0%.

Note: Display appropriate error messages wherever applicable.

Write a Python Program to implement the same.

```
bill_id = 1001
customer_id = 101
bill_amount = 200.0
discounted_bill_amount = 0.0
print("Bill Id: %d" %bill_id)
print("Customer Id: %d" %customer_id)
print("Bill Amount:Rs. %f" %bill_amount)
if ((customer_id > 100) and (customer_id <= 1000)) is True:
    if bill_amount >= 500:
        discounted_bill_amount = bill_amount - bill_amount * 10 / 100
        print("Discounted Bill Amount:Rs. %f" %discounted_bill_amount)
    else:
        print("No Discount")
else:
    print("Invalid Customer id, customer id must between 101 and 1000")
```

Note the usage of logical AND operator

Note the use of nested if statement

Note the display of appropriate error messages

Output:

```
Bill Id: 1001
Customer Id: 101
Bill Amount:Rs. 200.000000
No Discount
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have understood the implementation of operators and control structures using a real world problem

Assignment 15: Control Structures – Hands – on – Practice

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: The finance department of a company wants to calculate the monthly net pay of one of its employee by finding the income tax to be paid (in Indian Rupees) and the net salary after the income tax deduction. The employee should pay income tax if his monthly gross salary is more than Rs. 10,000 (Indian Rupees) and the percentage of income tax should be considered as 20% of the gross salary. Display the employee id, basic salary, allowances, gross pay, income tax and net pay.

Note:

Employee Id must be considered as 1001,

Basic salary of the employee must be considered as Rs.15000.00 and

Allowances must be considered as Rs.6000.00

Write a Pseudo code and Python program in Eclipse to solve the above real world problem.

Refer below for the formulae to be used.

Monthly Gross Salary = Basic Salary + Allowances

Net Salary = Gross Salary – Income Tax

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have understood the implementation of operators and control structures using a real world problem

Assignment 16: Control Structures – Hands - on - Practice

Objective: Given a real world problem be able to understand the need for control structures and operators to implement the logic and solve the problem

Problem Description: Extend the program written for Assignment 15 to find the income tax to be paid (In Indian Rupees) and the total salary after the income tax deduction as per the details given in below table.

Gross Salary (In Indian Rupees)	Income Tax percentage
Below 5,000	Nil
5,001 to 10,000	10 %
10,001 to 20,000	20%
More than 20,000	30%

Display the employee id, basic salary, allowances, gross pay, income tax and net pay.

Note:

Employee Id must be considered as 1001,

Basic salary of the employee must be considered as Rs.15000.00 and

Allowances must be considered as Rs.6000.00

Write a Pseudo code and Python program in Eclipse to solve the above real world problem.

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have understood the implementation of operators and control structures using a real world problem

Assignment 17: Iteration Control Structures - Guided Activity

Objective: Given a real world problem, implement the logic and solve the problem using appropriate constructs (sequential, selection, iteration) using an object oriented programming language (Python)

Problem Description:

Dry run the below code snippets and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Code – 1:

```
counter = 1

while counter <= 3:
    print(counter)
    counter += 1
print("End of Program")
```

Output:

```
1
2
3
End of Program
```

Code-2:

```
print("To find the sum of first 10 integers:")
result = 0
for value in range(1,11):
    result = result + value
print("Sum:",result);
```

Output:

```
To find the sum of first 10 integers:
Sum: 55
```

Code-3:

```
number = 1
result = 0
while number < 5:
    result = result + number
    number = number + 1
print(result)
```

Output:

```
10
```

Code-4:

```
result = 0
for index in range(40, 10, -2):
    if(index % 5 == 0):
        result = result + index
    print(result)
```

Output:

```
40
70
90
```

Code-5:

```
amount = 100.0
interest = 0.0
months = 1
while months < 6:
    interest = amount * 0.2
    amount = amount + interest
    months += 1
print(amount)
```

Output:

```
248.83200000000002
```

Estimated time: 20 minutes

Summary of this assignment: In this assignment, you have understood the implementation of iteration control structures.

Assignment 18: break statement - Demo

Objective: Given a real world problem, implement the logic and solve the problem using appropriate constructs (sequential, selection, iteration) using an object oriented programming language (Python)

Problem Description:

Dry run the below code snippet and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Code:

```
count = 0
result = 0
for count in range (1, 10):
    result = result + count
    if result > 6:
        break
print("Result =", result)
```

Note the use of break statement

On encountering the break statement, control will move out of the loop as shown

Output:

```
Result = 10
```

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have understood the implementation of break statement.

Assignment 19: continue statement - Demo

Objective: Given a real world problem, implement the logic and solve the problem using appropriate constructs (sequential, selection, iteration) using an object oriented programming language (Python)

Problem Description:

Dry run the below code snippet and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Code:

```
count = 0
for count in range(0,10):
    if 4 == count:
        continue
    print(count)
```

Note the use of continue statement

On encountering the continue statement, control will move to the increment portion of the for loop

Output:

```
0
1
2
3
5
6
7
8
9
```

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have understood the implementation of continue statement.

Assignment 20: Iteration Control Structure – Debugging - Guided Activity

Objective: Given a real world problem, implement the logic and solve the problem using appropriate constructs (sequential, selection, iteration) using an object oriented programming language (Python)

Problem Description: The code given below is written to display all the even numbers between 50 and 80 (both inclusive). Debug the program to get the correct output.

Step 1: Type the below program in Eclipse, save the file as for_loop.py, compile and execute.

```
for i in range (50, 80):  
    if i % 2 == 0:  
        print(i)  
    else:  
        break
```

Step 2: Correct the logical error in the code, save, compile and execute the code

Step 3: Implement the same logic using while loop

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of iteration control structure and break statement.

Strings - Assignments

Assignment 21: Strings – Observations from Retail Application - Guided Activity

Objective: Observe the need for features in the retail application scenario to correlate the application of Strings

Problem Description: In the retail application, along with the earlier details included for Customer, the retail shop wants to keep track of customer name. Customer name should be between 3 and 20 characters.

Answer the following question:

What do you think is needed to implement the solution for the above problem?

Estimated time: 5 minutes

Summary of this assignment: In this assignment, you have understood the need of strings using a retail application scenario

Assignment 22: Strings – Demo

Objective: Observe the need for features in the retail application scenario to correlate the application of Strings

Problem Description: In the retail application, along with the earlier details included for Customer, the retail shop wants to keep track of customer name. Customer name should be between 3 and 20 characters.

```
bill_id = 1001
customer_id = 1001
bill_amount = 2000
customer_name = Kevin
```

Code:

```
bill_id = input("Please enter Bill id")
customer_id = input("Please enter Customer id")
bill_amount = input("Please enter bill Amount")
customer_name = input("Please enter Customer Name")
if ((len(customer_name) >= 3) and (len(customer_name) <= 20)) is True:
    print("Bill Id: ",bill_id)
    print("Customer Id: ",customer_id)
    print("Bill Amount:Rs. ",bill_amount)
    print("Customer Name: ",customer_name)
else:
    print("Invalid customer name. Customer name must be between 3 and 20 characters");
```

Note how len() method is used for checking the number of characters in a String object

Output:

```
Please enter Bill id101
Please enter Customer id1001
Please enter bill Amount2000
Please enter Customer NameKevin
Bill Id: 101
Customer Id: 1001
Bill Amount:Rs. 2000
Customer Name: Kevin
```

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have understood the implementation of Strings and String methods using a retail application scenario

Assignment 23: Strings - Quiz

Objective: Revisit String concepts through a quiz

Problem Description:

Assume,
string1 = "Infosys Limited"
string2 = "Mysore"

Predict the output of following statements:

```
Q1: print(string1[:4])

Q2: print(string1[-1])

Q3: print(string1 * 2)

Q4: print(string1[:-1] + string2 + string1[:-1])

Q5: print (string2[1])

Q6: print (string2[4])

Q7: print(string1 * 2 + string1[:-1] + string2)
```

ANSWERS

```
Q1: Info
Q2: d
Q3: Infosys LimitedInfosys Limited
Q4: Infosys LimiteMysoreInfosys Limite
Q5: y
Q6: r
Q7: Infosys LimitedInfosys LimitedInfosys LimiteMysore
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have revisited concepts of strings through code snippets

Assignment 24: Strings – Hands - on - Practice

Objective: Given a computational problem, be able to use the right data structures (lists and strings) and implement the solution to the problem and test using a set of values in an IDE

Problem Description:

- Write a program to count and display the number of capital letters in a given string.
- Write a program to check if the given string is Palindrome or not?
- Write a program to count the number of each vowel in a string
- Write a program to remove all punctuation from the string provided by the user

punctuations = "'!()-[]{};:'\".,<>./?@#\$%^&*~_'"

Hint: Use membership operators IN, Not IN

Estimated time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Strings concept for the given computational problem.

Assignment 25: Strings – Hands – on - Practice

Objective: Given a computational problem, be able to use the right data structures (lists and strings) and implement the solution to the problem and test using a set of values in an IDE

Problem Description: Write a Python program to accept a string and display the resultant string in reverse order. The resultant string should contain all characters at the even position of accepted string ignoring blank spaces.

accepted_string: An apple a day keeps the doctor away

resultant_string: Aapedyteotrw

expected_output: ywrtoetpeydepaA

Estimated time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Strings concept for the given computational problem.

Assignment 26: Strings – Hands – on - Practice

Objective: Given a computational problem, be able to use the right data structures (lists and strings) and implement the solution to the problem and test using a set of values in an IDE

Problem Description: Given a string containing both upper and lower case letters. Write a Python program to count the number of repeated characters and display the maximum count of a character along with the character.

Sample Input: ABaBCbGc

Output:

2A

3B (three times B is repeated)

2C

1G

Estimated time: 10 minutes

Summary: In this assignment, you have understood the application and implementation of Strings concept for the given computational problem.

Assignment 27: Strings – Hands - on - Practice

Objective: Given a computational problem, be able to use the right data structures (lists and strings) and implement the solution to the problem and test using a set of values in an IDE

Problem Description: Consider 2 strings string1 and string2 and display the merged_string as the output. The merged_string should be capital letters from both the strings in the order they appear.

Note: Each character should be checked if it is a capital letter and then it should be merged.

Sample Input:

string1: I Like C

string2: Mary Likes Python

merged_string: ILCMLP

Estimated time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Strings concept for the given computational problem.

Assignment 28: Operations on Tuples - Demo

Objective: To understand the operations that can be performed on Tuples through a quiz

Problem Description:

Assume,
head = ("CEO",)
elements = ('Air', 'Water', 'Fire', 'Light', 'Land')

Predict the output of following statements:

```
Q1: print(head)

Q2: print(elements)

Q2: print(elements[3])

Q3: elements[0]='Ice'

Q5: print(elements[-1])

Q6: print(elements[4])

Q7: print(elements[5])
```

ANSWERS

```
Q1: ('CEO',)
NOTE: In this case the symbol , is mandatory without which it becomes
just a string assignment operation

Q2: ('Air', 'Water', 'Fire', 'Light', 'Land')
```

Q3: Light

Q4: Type Error

Note: Tuples are immutable. Cannot change the contents of tuple

Q5: Land

Q6: Land

Q7: Index Error: tuple index out of range

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have revisited concepts of tuples through code snippets

Assignment 29: Tuples – Hands – on - Practice

Objective: Given a List of elements representing a computational problem, be able to provide solution by performing required operations using data structures like tuple from an object oriented language (Python) using Eclipse IDE

Problem Description: Consider the list of courses opted by a Student “John” and available electives as a part of Student Management System.

courses = (“Python Programming”, “RDBMS”, “Web Technology”, “Software Engg”).
electives = (“Business Intelligence”, “Big Data Analytics”)

Write a Python Program to satisfy following business requirements:

- List the number of courses opted by Student by “John”
- List all the courses opted by Student “John”.
- Students “John” is also interested in elective course mentioned above. Print the updated tuple including electives
- Check whether Student “John” is allowed to change his course from “Software Engg” to “Computer Networks”. If yes, print the updated course list else mention the reason for the same.

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of Tuple operations for given business scenario.

Lists - Assignments

Assignment 30: Accessing Elements from Lists - Demo

Objective: Given a List of elements representing a computational problem, be able to access elements in different ways using an object oriented language (Python) using an IDE

Problem Description:

Assume,
language = ['Python']
languages = ['Python', 'C', 'C++', 'Java', 'Perl']

Predict the output of following statements:

```
Q1: print(language)

Q2: print(languages)

Q3: print(languages[0:3])

Q4: print(languages[1:4])

Q5: print(languages[2])

Q6: print(languages[5])

Q7: print (languages[0] + " and " + languages[1] + " are quite
different!")

Q8: print ("Accessing the last element of the list: " + languages[-1])
```

ANSWERS

```
Q1: ['Python']

Q2: ['Python', 'C', 'C++', 'Java', 'Perl']
```

Q3: ['Python', 'C', 'C++']

Q4: ['C', 'C++', 'Java']

Q5: C++

Q6: Index Error: list index out of range

Q7: Python and C are quite different!

Q8: Accessing the last element of the list: Perl

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have revisited concepts of lists through code snippets

Assignment 31: Lists – Hands - on - Practice

Objective: Given a computational problem, be able to use the right data structures (lists and strings) and implement the solution to the problem and test using a set of values in an IDE

Problem Description: Write a Python program to generate first 'n' Fibonacci numbers. Store the generated Fibonacci numbers in a list and display it.

Sample input: Enter n 5

Sample Output: List: [0, 1, 1, 2, 3]

Estimated time: 10 minutes

Summary: In this assignment, you have understood the application and implementation of Lists concept for the given computational problem.

Assignment 32: Lists – Hands - on - Practice

Objective: Given a List of elements representing a computational problem, be able to sort the elements in ascending/descending order using an object oriented language (Python) using an IDE

Problem Description: You have a bundle of currency of varied denominations. You want to arrange them in descending order.

Given below is one approach to perform the above operation.

Algorithm: Bubble Sort (List of N element)

Input: A List of N elements to be sorted

Output: A List of sorted (increasing order) of N elements

Step 1: Repeat steps 2 and 3 for n-1 times

Step 2: Repeat step 3 for 1 time less than n

Step 3: Compare first element with next element and swap if second element is greater

Pseudo-code is represented below:

```
1.   for i = 1 to N - 1
2.       for j = 1 to N - i
3.           if ( list[ j ] > list[ j + 1 ]) then
4.               interchange (list[ j ] , list[ j + 1 ])
5.           end-if
6.       end-for
7.   end-for
```

Write a Python program to sort the elements using bubble sort technique in the list and display the elements in descending order.

Sample Input: [4, 1, 6, 2, 8, 5]

Sorted: [1, 2, 4, 5, 6, 8]

Output: [8, 6, 5, 4, 2, 1]

Estimated time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Lists concept for the given computational problem.

Assignment 33: Lists - Hands - on - Practice

Objective: Given a computational problem, Implement the solution for the problem using suitable data structures from an object oriented language (Python) using an IDE

Problem Description: ABC Retail Store sells different varieties of Furniture to the customers. The list of furnitures available and its cost list are given below:

Furniture	Sofa set	Dining table	T.V. Stand	Cupboard
Cost in Rs.	20,000	8,500	4,599	13,920

The furniture's and its corresponding Cost should be stored as a list. If the required furniture is available in list of furniture's listed above and Quantity purchased is greater than zero, only then bill amount should be calculated. In case of invalid values for furniture required by the customer and quantity purchased, consider bill amount to be 0.

Initialize required furniture and required quantity with different values and test the results.

Write a Python program to calculate and display the bill amount to be paid by the customer based on the furniture bought and quantity purchased.

Estimated time: 15 minutes

Summary: In this assignment, you have understood the application and implementation of Lists concept for the given computational problem.

Assignment 34: Sets - Demo

Objective: Given a Set of elements representing a computational problem, be able to perform different operations using an object oriented language (Python) using an IDE

Problem Description:

Consider the sets,

```
fruits = {"apple", "orange", "banana", "apple", "pear", "papaya", "papaya"}
```

```
fruit_basket = {"apple", "banana", "grapes", "mango", "kiwi"}
```

Predict the output of following statements:

```
Q1: print(fruits)
```

```
Q2: print(fruits & fruit_basket)
```

```
Q3: print(fruits | fruit_basket)
```

```
Q4: print(fruits - fruit_basket)
```

```
Q5: print(fruits ^ fruit_basket)
```

```
Q6: print(len(fruit_basket))

Q7: print("pear" in fruits)

Q8: print("pear" not in fruit_basket)

Q9: print(fruits.issubset(fruit_basket))

Q10: print(fruits.issuperset(fruit_basket))

Q11: print(fruit_basket.copy())
```

ANSWERS

```
Q1: {'banana', 'apple', 'orange', 'papaya', 'pear'}

Q2: {'apple', 'banana'}

Q3: {'banana', 'mango', 'orange', 'apple', 'grapes', 'papaya', 'kiwi',
'pear'}

Q4: {'orange', 'papaya', 'pear'}

Q5: {'mango', 'orange', 'grapes', 'kiwi', 'papaya', 'pear'}

Q6: 5

Q7: True

Q8: True

Q9: False

Q10: False

Q11: {'mango', 'apple', 'grapes', 'kiwi', 'banana'}
```

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have revisited concepts of Sets through code snippets

Assignment 35: Sets – Hands - on - Practice

Objective: Given a Set of elements representing a computational problem, be able to provide solution by performing required operations on sets from an object oriented language (Python) using an IDE

Problem Description: Consider the scenario from course in student management system. Given below are 2 Sets representing the names of students enrolled for a particular course.

```
java_course = {"John", "Jack", "Jill", "Joe"}  
python_course = {"Jake", "John", "Eric", "Jill"}
```

Write a Python program to satisfy below mentioned business requirements:

- List the number of Students enrolled for Python course
- List the names of Students enrolled for Java course only
- List the names of Students enrolled for Python course only
- List the number and names of Students enrolled for both Java and Python courses
- List the number and names of Students enrolled for either Java or Python courses but not both
- List names and number of Students enrolled for either Java or Python courses

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of Set operations for given business scenario.

Assignment 36: Dictionary - Demo

Objective: Given a Dictionary of elements representing a computational problem, be able to perform different operations using an object oriented language (Python) on Eclipse IDE

Problem Description:

Given below is a Dictionary customer_details representing customer Details from Retail Application - Customer Id is key and Customer Name is value.

```
customer_details = { 1001 : "John", 1004 : "Jill", 1005: "Joe", 1003 : "Jack" }
```

Write Python code to perform below mentioned operations:

- Print details of Customers

- b. Print number of Customers
- c. Print Customer names in ascending order
- d. Delete the details of customer with customer id = 1005 and print updated dictionary
- e. Update the name of customer with customer id = 1003 to "Mary" and print updated dictionary
- f. Check whether details of customer with customer id 1002 exists in the dictionary.

ANSWERS

```
a: print(customer_details)

b: print(len(customer_details))

c: print(sorted(customer_details.values()))

d: del(customer_details[1005])
   print(customer_details)

e: customer_details[1003] = "Mary"
   print(customer_details)

f: print(1002 in customer_details)
```

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of Dictionary operations for given business scenario.

Assignment 37: Dictionary – Hands - on - Practice

Objective: Given a computational problem, select the right set of data structures (lists and dictionary) and implement the solution to problem and test using a set of values in an IDE

Problem Description: Consider the scenario of processing marks of students for a course in student management system. Given below is the list of marks scored by students. Find top three scorers for the course and also display average marks.

Implement the solution for given business scenario.

Student Name	Marks Scored
John	86.5
Jack	91.2
Jill	84.5
Harry	72.1
Joe	80.5

Estimated Time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Dictionary concept for the given computational problem.

Assignment 38: Functions – Pass by Reference - Demo

Objective: Given a computational problem, implement functions and solve it using parameter passing technique followed in python

Problem Description: Consider the distance in miles between two locations:

- distance between Phoenix, Arizona and Salt Lake City, Utah in USA
- distance between Phoenix, Arizona and Tampa, Florida in the US.

We want to compare the distances and check which one is far off from Phoenix, Arizona.

Provided below are codes written to solve the above problem using pass by reference technique – using Required arguments and Keyword Arguments.

Code:

Method 1: Pass by Reference – Required Arguments

```
def compare(phoenix_to_slc, phoenix_to_tampa):  
    if phoenix_to_slc > phoenix_to_tampa:  
        print("SLC is far from Phoenix compared to Tampa, Florida")  
  
    elif(phoenix_to_slc < phoenix_to_tampa):  
        print("Tampa, Florida is far from Phoenix compared to SLC")  
  
    else:  
        print("Both locations are equidistance from Phoneix")
```

Phoenix_to_tampa is formal argument


```
compare (1790, 506)
```

506 is the actual argument which is passed and assigned to the formal argument, phoenix_to_tampa, when the method compare() is invoked. It is mandatory to pass both the arguments

Method 2: Pass by Reference – Keyword Arguments

```
def compare(phoenix_to_slc, phoenix_to_tampa):  
    if phoenix_to_slc > phoenix_to_tampa:  
        print("SLC is far from Phoenix compared to Tampa, Florida")  
  
    elif(phoenix_to_slc < phoenix_to_tampa):  
        print("Tampa, Florida is far from Phoenix compared to SLC")  
  
    else:  
        print("Both locations are equidistance from Phoneix")  
  
compare (phoenix_to_tampa = 506, phoenix_to_slc = 1790)
```

Output:

Here parameters are passed by keyword arguments. Hence, it is not mandatory to pass both the arguments and need not follow positional order

```
SLC is far from Phoenix compared to Tampa, Florida
```

Estimated Time: 20 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of functions and parameter passing technique - pass by reference through keyword arguments and reference arguments

Assignment 39: Functions – Pass by Reference – Hands - on - practice

Objective: Given a computational problem, Implement functions and solve it using appropriate parameter passing techniques (pass by reference)

Problem Description: At an airport, a traveler is allowed entry into the flight only if he clears the following checks

- i. **Baggage Check**
- ii. **Immigration Check**
- iii. **Security Check**

The logic for the check methods are given below:

Implementation details

check_baggage (baggage amount)

- Check if baggage_amount is greater than or equal to 0 and less than or equal to 40.
- If *baggage_amount* is VALID
 - return TRUEelse
 - return FALSE

check_immigration (expiry_year)

- Check if expiry_year is greater than or equal to 2001 and less than or equal to 2025.
- If *expiry_year* is VALID
 - return TRUEelse
 - return FALSE

check_security(noc_status):boolean

- If *noc_status* is TRUE
 - return TRUEelse
 - return FALSE

traveler()

In *traveler()* function, initialize the traveler Id and traveler name and invoke the functions *check_baggage()*, *check_immigration()* and *check_security()* by passing required arguments. Refer the table below for values of arguments.

Variable	Value
<i>traveler_id</i>	1001
<i>traveler_name</i>	Jim
<i>baggageAmount</i>	35
<i>expiryDate</i>	2019
<i>nocStatus</i>	true

If all values of *check_baggage()*, *check_immigration()* and *check_security()* are **true**,
display *traveler_id* and *traveler_name*
display "Allow Traveller to fly!"
else,

display *traveler_id* and *traveler_name*
display "Detain Traveller for Re-checking!"

Estimated time: 25 minutes

Summary of this assignment: In this assignment, you have learnt pass by reference technique

Assignment 40a: Recursion – Demo

Objective: Given a problem, be able to solve the problem by representing the logic using pseudo code representation.

Problem Description: Consider the problem of finding factorial of a number. The same can be implemented in Python using Recursion.

Consider the following pseudo-code:

FACTORIAL (number)

1. **if** (number = 0) **then**
2. **return**(1)
3. **else**
4. **return**(number * **factorial** (number - 1))
5. **end-if**

Note the name of the function factorial and the argument passed – number whose factorial is to be calculated

Note the termination condition

Note that the factorial() is making a call to itself with one less the number

The same can be represented in Python:

#Factorial function calling itself recursively

```
def factorial (number):
```

```
    if number == 0:
```

```
        return 1
```

```
    else:
```

```
        return number * factorial(number - 1)
```

#Calling factorial function

```
raw_input = input("Enter the number to compute factorial: ")
```

```
number = int(raw_input)
```

```
result = factorial (number)
```

Note the conversion of String input to Integer type

#Print the factorial number

```
print("The factorial of ",number, " is ", result)
```

Output:

```
Enter the number to compute factorial: 4
The factorial of 4 is 24
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt how to implement recursion.

Assignment 40b: Recursion – Hands – on - Practice

Objective: Given a problem, be able to solve the problem by representing the logic using pseudo code representation.

Problem Description: Consider the problem of generating Fibonacci series using Recursion. The same can be implemented in Python using Recursion.

Consider the following pseudo-code:

FIBO (number)

1. **if** (number = 0) **then**
2. **return** (0)
3. **else if** (number = 1) **then**
4. **return** (1)
5. **else**
6. **return** FIBO(number - 1) + FIBO(number - 2)
7. **endif**

Write a program in Python to implement the same using Recursion and execute it in Eclipse. Print appropriate error messages if the user enters negative number as input.

Note: Same problem has been discussed under Lists using iteration – Assignment No.31

Estimated time: 20 minutes

Summary of this assignment: In this assignment, you have learnt how to implement recursion.

Assignment 41: Recursion – Hands – on - Practice

Objective: Given a problem, be able to solve the given computational using Recursion.

Problem Description:

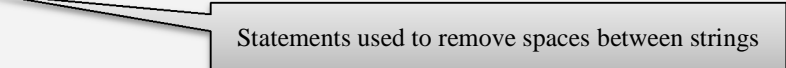
Write a program in Python to check whether given string is palindrome or not using Recursion.

```
#Palindrome function calling itself recursively
def is_palindrome(given_string):
    #Remove spaces in given strings
    S=given_string.split()
    S = ''.join(S)

    if len(S)==1 or len(S)==0:
        return True
    else:
        # -1 index refers to last element in list/string
        if S[0] == S[-1] and is_palindrome(S[1:-1]):
            return True
        else:
            return False

#Calling palindrome function
str1 ="AB  BA"
str2 = "abcdeedcba"
print("%s is palindrome:%s"%(str1,is_palindrome(str1)))
print("%s is palindrome:%s"%(str2,is_palindrome(str2)))

str3="Mysore"
print("%s is palindrome:%s"%(str3,is_palindrome(str3)))
```



Output:

```
AB  BA is palindrome: True
abcdeedcba is palindrome: True
Mysore is palindrome: False
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt how to implement recursion for computational problems.

Assignment 42: Recursion – Hands – on - Practice

Objective: Given a problem, be able to solve the given computational using Recursion.

Problem Description:

Write a program in Python to reverse given string using Recursion.

```
#Reverse string function calling itself recursively
def rev_string(given_string):
    if(len(given_string)<=1):
        return given_string
    return rev_string(given_string[1:]) + given_string[0]

#Calling reverse string function
print(rev_string("Hello"))
print(rev_string("Mississippi"))
```

Output:

```
Reverse of string "Hello" is olleH
Reverse of string "Mississippi" is ippississim
```

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have learnt how to implement recursion for computational problems.

Assignment 43: Strings built - in functions - Guided Activity

Objective: Revisit String built-in functions through match the following.

Problem Description:

There are many built-in String methods available in Python. Match each method to what it does.

Method	What the method does
text.endswith(".jpg")	Return a copy of the string with all occurrences of one substring replaced by another
text.upper()	Return a copy of the string converted to lowercase
text.lower()	Return the value True if the string has the given substring at the beginning

<code>text.replace("tomorrow", "Saturday")</code>	Return the value True if the string has the given substring at the end
<code>text.strip()</code>	Returns the first index value when the given substring is found
<code>text.find("python")</code>	Return a copy of the string with the leading and trailing whitespace removed
<code>Text.startswith("<HTML>")</code>	Return a copy of the string converted to uppercase

ANSWERS:

Method	What the method does
<code>text.endswith(".jpg")</code>	Return the value True if the string has the given substring at the end
<code>text.upper()</code>	Return a copy of the string converted to uppercase
<code>text.lower()</code>	Return a copy of the string converted to lowercase
<code>text.replace("tomorrow", "Saturday")</code>	Return a copy of the string with all occurrences of one substring replaced by another
<code>text.strip()</code>	Return a copy of the string with the leading and trailing whitespace removed
<code>text.find("python")</code>	Returns the first index value when the given substring is found
<code>Text.startswith("<HTML>")</code>	Return the value True if the string has the given substring at the beginning

Estimated time: 10 minutes

Summary of this assignment: In this assignment, you have revisited String built-in functions through match the following.

Assignment 44: Lists built-in functions - Guided Activity

Objective: Given a computational problem, select the right set of data structures (lists and dictionary) and implement the solution to problem and test using a set of values in an IDE

Problem Description: Consider the price list of various items in the Retail Store. Customer John wants to know the

- i) Price of Costliest item sold in Retail store
- ii) Average price of items in the Retail store
- iii) Display of item price list in increasing order

Implement the solution using user-defined and built-in functions to accomplish above mentioned business requirement.

Code:

```
item_price = [1050, 2200, 8575, 485, 234, 150, 399]

def price_max(item_price):
    print("Price of Costliest item in the Retail Store: ",max(item_price))

def average_price(item_price):
    total_price = 0
    for i in item_price:
        total_price += i
    average_price = total_price/len(item_price)
    print("Average Price of items in the Retail Store: ",average_price)

def display_sorted_price(item_price):
    item_price.sort()
    print("Item Price List in increasing order: ",item_price)

price_max(item_price)
average_price(item_price)
display_sorted_price(item_price)
```

Output:

```
Price of Costliest item in the Retail Store: 8575
Average Price of items in the Retail Store: 1870.4285714285713
Item Price List in increasing order: [150, 234, 399, 485, 1050, 2200, 8575]
```

Estimated Time: 20 minutes

Summary: In this assignment, you have understood the application and implementation of Lists concept for the given computational problem.

Assignment 45: Lists – Stack – Observation from real world - Guided Activity

Objective: Given a problem scenario, be able to identify the way in which the data is stored in the computer memory

Problem description: Access your online banking application.
(For Example, the ICICI application at the following link)

<http://www.icicibank.com/safe-online-banking/safe-online-banking.html>

Once you login to your account select Mini statement. A statement of the last five/Ten transactions performed will be displayed on your screen

1. What is the order in which the transactions are displayed?
2. How are the previous transactions stored in the memory?

Estimated Time: 5 minutes

Summary of this assignment: In this assignment you have learnt that there are different ways in which we can store data in the computer memory for processing.

Assignment 46: Lists – Stack - Demo

Objective: Given a data structure, be able to identify the operations to be performed and implement the solution.

Problem Description: In stack, the elements are removed in reverse order of their insertion. So, stack is called Last-In-First-Out (LIFO) List. Stack can be implemented using a list.

Assumptions: List of size n is used to implement stack. List index starts from 0. Stack is referred using S

Algorithm: ISEMPY

Input: A stack

Output: True or false (check for stack is empty or not)

1. Check whether it is empty list, if yes return true.
2. Otherwise return false.

Pseudo-code is represented below:

IS_EMPTY(S)

1. **if** (S = []) **then**
2. **return** True
3. **else**
4. **return** False
5. **end-if**

Algorithm: ISFULL**Input: A stack****Output: True or false (check for stack is full or not)**

1. Check the length of list = size of stack return true.
2. Otherwise return false.

Pseudo-code is represented below:**IS_FULL(S)**

1. **if** (len(S) = size) **then**
2. **return** True
3. **else**
4. **return** False
5. **end-if**

Algorithm: PUSH**Input: A stack and an element to be inserted****Output: Stack after insertion**

1. Check for overflow condition, if ISFULL returns true then,
 - 1.1. Display "stack is full" and exit.
2. Otherwise
 - 2.1. Insert the element to end of list

Pseudo-code is represented below:**STACK_PUSH (S, new element)**

1. **if** (ISFULL()) **then**
2. **display** "stack is full"
3. **return**
4. **else**
5. S.append(new element)
6. **return** S
7. **end-if**

Algorithm: POP**Input: A stack****Output: Deleted element**

1. Check for underflow condition, if ISEMPTY returns true then,
 - 1.1. Display "Stack is empty" and exit.
2. Otherwise
 - 2.1. Pop the element at end of list
 - 2.2. Return the element deleted

Pseudo-code is represented below:**STACK_POP**

1. **if** (ISEMPTY()) **then**
2. **display** "stack is empty"
3. **return**
4. **else**
5. item = S.pop()
6. **return** item
7. **end-if**

The above pseudocode can be implemented and executed in Python using some built-in methods as shown below:

```
list_size = 3
s=[]

def push(s, item):
    if len(s) == list_size:
        print("Stack Overflow")
    else:
        s.append(item)
        print("Pushed element",item,"to Stack S")
        display_elements()

def pop():
    item = 0
    print("Pop Operation")
    if len(s) == 0:
        print("Stack Underflow")
    else:
```

Call the display_elements() method to display all the elements in stack

```
        item = s.pop()
        print("Element",item,"is popped")

def display_elements():
    if len(s) > 0:
        print("Elements in Stack:",s)
    else:
        print("Stack is empty")

push(s, 12)
push(s, 24)
push(s, 70)
push(s, 28)
display_elements()
pop()
pop()
push(s, 15)
pop()
pop()
display_elements()
pop()
```

Output:

```
Pushed element 12 to Stack S
Elements in Stack: [12]
Pushed element 24 to Stack S
Elements in Stack: [12, 24]
Pushed element 70 to Stack S
Elements in Stack: [12, 24, 70]
Stack Overflow
Elements in Stack: [12, 24, 70]
Pop Operation
Element 70 is popped
Pop Operation
Element 24 is popped
Pushed element 15 to Stack S
Elements in Stack: [12, 15]
Pop Operation
Element 15 is popped
Pop Operation
```

```
Element 12 is popped  
Stack is empty  
Pop Operation  
Stack Underflow
```

Estimated time: 15 minutes

Summary of this assignment: In this assignment, you have learnt how to implement stack using built-in methods of list.

Assignment 47: Lists – Stack - Hands - on - Practice

Objective: Given a data structure, be able to identify the operations to be performed.

Problem description: In stack, the elements are removed in reverse order of their insertion. So, stack is called Last-In-First-Out (LIFO) List. Stack can be implemented using either an array or a linked list.

Construct a stack containing the elements 12, 30, 45, 15. The top element in the stack is 15. Use built-in functions to push an element 35 to stack and pop elements 45, 15.

Display appropriate error messages for stack overflow and stack underflow scenario.

Assumptions: List of size 3 is used to implement stack. List index starts from 0.

Estimated Time: 10 minutes

Summary of this assignment: In this assignment you have understood the need for Queue data structure.

Assignment 48: Lists – Queue - Guided Activity

Objective: Given a data structure, be able to identify the operations to be performed.

Problem description:

Background: In queue, elements are removed in the same order, in which they were inserted. So, queue is called First-In-First-Out (FIFO) List. Queue can be implemented using list.

Assumptions: List of size n is used to implement queue. List index starts from 0. Queue is referred using Q. Q.front refers to front of the queue and Q.rear refers to end of the queue.

Algorithm: ENQUEUE**Input: A queue and an element to be inserted****Output: Queue after insertion**

1. Check whether queue is full or not, if rear index = n then,
 - 1.1. Display "Queue is full" and exit.
2. Otherwise
 - 2.1. Place the value at rear index and exit.

Pseudo-code is represented below:**ENQUEUE(Q, new element)**

1. **if** (Q.rear = n) **then**
2. **display** "Queue is full"
3. **return** Error Code
4. **else**
5. Q.queue[rear] = new element
6. Q.rear = Q.rear+1
7. **return** Q

Algorithm: DEQUEUE**Input: A queue****Output: Removed element**

1. Check whether queue is empty or not, if rear index = front index then,
 - 1.1. Display "Queue is empty" and exit.
2. Otherwise
 - 2.1. Place the front index value in a variable Item
 - 2.2. Increment the front index by 1 and return Item.

Pseudo-code is represented below:**DEQUEUE (Q)**

1. **if** (Q.front = Q.rear) **then**
2. **display** "Queue is empty"
3. **return**
4. **else**
5. item = Q.queue[front]
6. Q.front = Q.front+1
7. **return** item
8. **end-if**

The above pseudocode can be implemented and executed in Python as shown below:

```
global max_size, elements, rear, front
max_size = 5
elements = [None] * max_size
rear = - 1
front = 0

def is_full():
    global max_size, elements, rear, front
    if(rear == max_size - 1):
        return True
    return False

def is_empty():
    global max_size, elements, rear, front
    if(front > rear):
        return True
    return False

def enqueue(data):
    global max_size, elements, rear, front
    if(is_full()):
        print("Queue is full!!!")
    else:
        rear += 1
        elements[rear] = data
        print("Inserted element at front end: ",data)

def dequeue():
    global max_size, elements, rear, front
    if(is_empty()):
        print("Queue is empty!!!")
    else:
        data = elements[front]
        front += 1
        print("Deleted element from rear end: ",data)
        return data

def display():
    global max_size, elements, rear, front
    print("Elements in Queue")
```

```
    for index in range(front, rear + 1):
        print(elements[index])

dequeue() #Queue is empty
enqueue(2)
enqueue(7)
enqueue(9)
display() #prints 2, 7, 9
dequeue() #delete an element '2' from front end
display() #prints 7, 9
enqueue(4)
enqueue(6)
display() #prints 7, 9, 4, 6
enqueue(5) #prints "Queue full"
dequeue() #delete an element '7' from front end
display() #prints 9, 4, 6
enqueue(5) #prints "Queue full"
```

Observe deletion of element 2 from end

Observe deletion of element 7 from end

Output:

```
Queue is empty!!!
Inserted element at front end:  2
Inserted element at front end:  7
Inserted element at front end:  9
Elements in Queue
2
7
9
Deleted element from rear end:  2
Elements in Queue
7
9
Inserted element at front end:  4
Inserted element at front end:  6
Elements in Queue
7
9
4
6
Queue is full!!!
Deleted element from rear end:  7
```



```
Elements in Queue
9
4
6
Queue is full!!!
```

Estimated Time: 10 minutes

Summary of this assignment: In this assignment you have understood the operations performed on Queue data structure

Assignment 49: Regular Expression – match() - Guided Activity

Objective: Discuss the usage of Regular Expression concepts through Code Snippets

Problem Description:

Dry run the below code snippets and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Assume *re* module is imported for the following code snippets

Code – 1:

```
sentence = 'we are humans'
matched = re.match(r'(.*) (.*) (.*)', sentence)
print(matched.groups())
```

- a. ('we', 'are', 'humans')
- b. (we, are, humans)
- c. ('we', 'humans')
- d. 'we are humans'

Output:

```
('we', 'are', 'humans')
```

Explanation: Function returns all the subgroups that have been matched

Code – 2:

```
sentence = 'we are humans'
matched = re.match(r'(.*) (.*) (.*)', sentence)
print(matched.group())
```

- a. ('we', 'are', 'humans')
- b. (we, are, humans)
- c. ('we', 'humans')
- d. we are humans

Output:

we are humans

Explanation: Function returns the entire match

Code – 3:

```
sentence = 'we are humans'
matched = re.match(r'(.*) (.*) (.*)', sentence)
print(matched.group(1))
```

- a. ('are', 'humans')
- b. are
- c. ('we', 'humans')
- d. we

Output:

we

Explanation: Function returns first element from matched group

Code – 4:

```
sentence = 'Dogs and Cats are Domestic Animals. But, I love Rabbits'
matched = re.match(r'Cats', sentence)
print(matched.group())
```

- a. Cats
- b. Dogs
- c. Error
- d. None of the above

Output:

Error

Explanation: Because match() finds matches only at the beginning of string

Estimated Time: 20 minutes

Summary of this assignment: In this assignment, you have revisited concepts of Regular Expressions through code snippets

Assignment 50: Regular Expression – findall() - Guided Activity

Objective: Discuss the usage of Regular Expression concepts through Code Snippets

Problem Description:

Dry run the below code snippets and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Assume *re* module is imported for the following code snippets

Code – 1:

```
sentence = 'Dogs and Cats are Domestic Animals. Dog is a faithful animal'
print(re.findall(r'dog', sentence))
```

- a. []
- b. Dog
- c. Error
- d. Dogs

Output:

```
[]
```

Explanation: Because findall() does case sensitive search

Code – 2:

```
sentence = 'Dogs and Cats are Domestic Animals. Dog is a faithful animal'
print(re.findall(r'Dog', sentence))
```

- a. []
- b. ['Dogs', 'Dog']
- c. Error
- d. ['Dog', 'Dog']

Output:

```
['Dog', 'Dog']
```

Explanation: Because findall() prints only the matched string

Estimated Time: 10 minutes

Summary of this assignment: In this assignment, you have revisited concepts of Regular Expressions through code snippets

Assignment 51: Regular Expression – search() and replace() - Guided Activity

Objective: Discuss the usage of Regular Expression concepts through Code Snippets

Problem Description:

Dry run the below code snippets and predict the output. You may want to confirm the output by executing the code using Eclipse IDE.

Assume *re* module is imported for the following code snippets

Code – 1:

```
sentence = 'Dogs and Cats are Domestic Animals. But, I love Rabbits'
matched = re.search(r'Cat', sentence)
print(matched.group())
```

- a. Cats
- b. Dogs
- c. Error
- d. Cat

Output:

Cat

Explanation: Because search() does not restrict matching to only beginning of string

Code – 2:

```
sentence = 'Dogs and Cats are Domestic Animals. Dog is a faithful animal'
matched = re.search(r'Dog', sentence)
print(matched.group())
```

- a. Dog
- b. Dogs
- c. ['Dog', 'Dog']
- d. ['Dogs', 'Dog']

Output:

Dog

Explanation: Because search() restricts its check to first occurrence of the string to be matched

Code – 3:

```
sentence = 'Dogs and Cats are Domestic Animals. Dog is a faithful animal'
matched = re.search(r'dog', sentence)
print(matched.group())
```

- a. Dogs
- b. Dog
- c. Error
- d. ['Dogs', 'Dog']

Output:**Error**

Explanation: Because search() does case sensitive search

Code – 4:

```
sentence = 'Dogs and Cats are Domestic Animals. But, I love Rabbits'
replaced_sentence = re.sub('Rabbits', 'Peacock', sentence)
print(replaced_sentence)
```

- a. Dogs and Cats are Domestic Animals. But, I love peacock
- b. Dogs and Cats are Domestic Animals. But, I love Peacock
- c. Error
- d. Dogs and Cats are Domestic Animals. But, I love Rabbits

Output:

Dogs and Cats are Domestic Animals. But, I love Peacock

Explanation: After successful search, sub() substitutes matched string with specified string.

Estimated Time: 15 minutes

Summary of this assignment: In this assignment, you have revisited concepts of Regular Expressions through code snippets

Assignment 52: Regular Expression - Hands - on - Practice

Objective: Given a computational problem, learn the usage of re module to provide the solution for the same.

Problem Description: Consider below scenario from Student Management System

- a. Accept student_id and validate whether it contains only digits
- b. If student_id is valid, accept student_name from the user and validate whether it contains only alphabets
- c. If student_name is valid, accept fees_amount paid by the student
 - Decimal point is optional in fees_amount
 - Only 2 digits are allowed after decimal point
- d. If invalid data is entered in any of the above steps, display appropriate error messages. Else, create an email_id for student as [student_name@ABC.com](#). Where ABC is the name of the college. Assume there are no duplicate names.
- e. Perform above validations using Regular Expressions and print details of Student: student_id, student_name, fees_amount, email_id

Estimated time: 20 minutes

Summary of this assignment: In this assignment, you have learnt the implementation of solution for a given business scenario using re module.