A

SUMMER TRAINING REPORT

ON

FANTASY CRICKET LEAGUE

INTERNSHALA

Programming with Python

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF COMPUTER APPLICATION DEPARTMENT OF COMPUTER APPLICATION

COLLEGE OF INNOVATTIVE MANAGEMENT AND SCIENCE

(CIMS)

Affiliated to University of Lucknow, Lucknow Uttar Pradesh

Session 2019-2020

SUBMITTED TO:
Mr. Atul Singh
(Asst. Prof.)

SUBMITTED BY:
Lucky Verma
Roll No. 181730625016
(BCA 5th Semester)

DECLARATION

I hereby declare that this submission is our own work and that to the best of our knowledge and beliefs. It contains no material previously published or written by neither any person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Lucky Verma
Roll No. 181730625016
(BCA 5th Semester)

ACKNOWLEDGEMENT

I wish to express my most sincere thanks to Sales Manager Internshala for providing me the opportunity to work with the renowned organization and enabling me to gain practical knowledge.

A special note of thanks is also reserve to, training and placement officer of **"COLLEGE OF INNOVATTIVE MANAGEMENT AND SCIENCE (CIMS)"** Moreover I am also in debt to my faculty guide Mr. Atul Mishra**,** for their kind help and co-operation for completing this project work.

I indeed thanks to Internshala for giving me an opportunity to have experience in a professionally run organization.

I would like to thank my parents and my teachers for taking me to this stage in life; it was their blessings which always gave me courage to face all challenges and made my path easier.

Lucky Verma
Roll No. 181730625016
(BCA 5th Semester)

TABLE OF CONTENT

1. **INTRODUCTION**
   - Python
   - History of python
   - Python features
   - Python installation
   - Basics of programming in python
   - Operators
   - Collection in python
   - Functions in python
   - Python modules
   - Python object oriented
   - Python mysql database access
   - Designing gui applications using pyqt in python
   - Database design
2. **PROJECT SCENARIO**
   - Problem Statement
   - Sample of Rules
   - Database Design
3. **CODING**
   - My Python Game Main Code
   - Calculate Points
   - cricket_db
   - Evaluate
   - New Team
   - Open
   - scoreboard
4. **RESULT OF PYTHON PROJECT**
5. **FUTURE SCOPE**
6. **REFERENCE**

INTRODUCTION

The online training, Programming with Python, is a 6-week training program covering essential concepts on the building blocks of Python, object-oriented programming, the use of SQLite database and development of GUIs for Python applications.

PYTHON

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

PYTHON INSTALLATION

From source

You can install Python from its source code if you want to really customize the binaryby setting certain options or flags during the build process.

1. Download the source from
   https://www.python.org/ftp/python/3.6.2/Python-3.6.2rc1.tgz
2. Type the following commands to extract and install Python from itssource:

   ```
   $sudo tar xzf python-3.6.2.tar.xz $./configure
   $sudo make install
   ```

3. To verify the installation, in the terminal type:

   ```
   $ Python3
   ```

4. The Python prompt (>>>) will appear.

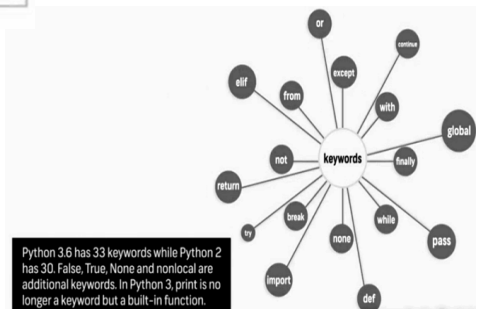
Using a wizard-based installer

1. Go to the Python website,
   https://www.python.org/ftp/python/3.6.2/python-3.6.2-macosx10.6.pkgand download the required version.
2. Run the downloaded file and follow the instructions in the installationwizard.

Basic Syntax

What is Syntax?

Computers are machines that can interpret your instructions only if you type them in the exact format that the computer expects. This expected format – the spelling, formatting and grammar – is called the syntax.



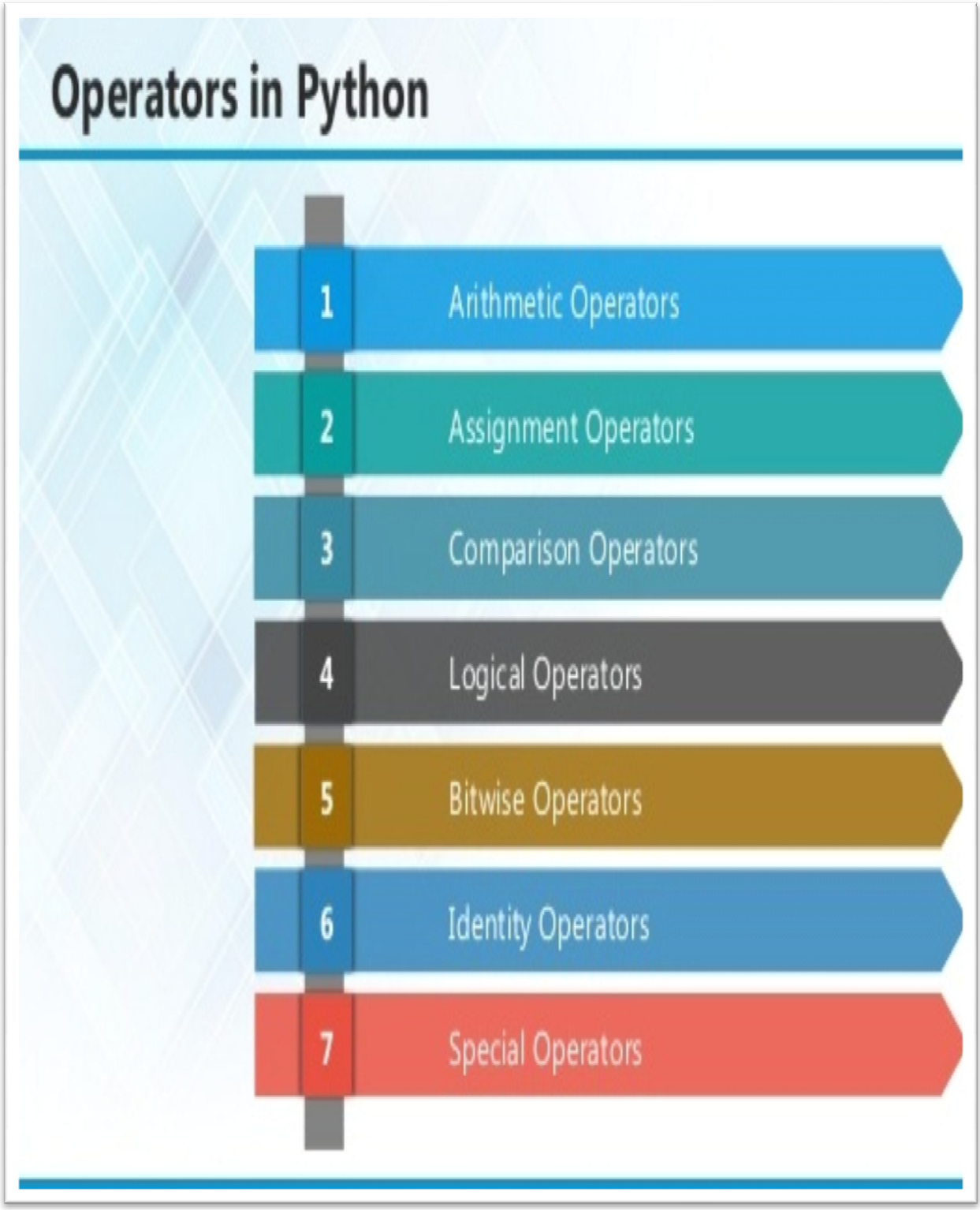
helloworld.py - C:/Python36/helloworld.py (3.6.2)

File Edit Format Run Options Window Help

```
#The purpose of this program is to display some text.

print ("Hello World")
print ("Welcome to Internshala") #Author Internshala.


#Comment 1
#Comment 2

##COmment 3
##Comment 4
##Comment 5

'''This is a multiline
string of
comments.'''
```

BASICS OF PROGRAMMING IN PYTHON

OPERATORS



ARITHMETIC OPERATORS

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

ASSIGNMENT OPERATOR

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a

%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

IDENTITY OPERATOR

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y

COMPARISON OPERATOR

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

LOGICAL OPERATOR

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Membership Operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Operators Precedence

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division

+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

COLLECTION IN PYTHON

LIST

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Built-in List Functions & Methods:

Python includes the following list functions −

SN	Function with Description
1	**cmp(list1, list2)** Compares elements of both lists.
2	**len(list)** Gives the total length of the list.
3	**max(list)** Returns item from the list with max value.
4	**min(list)** Returns item from the list with min value.

5	**list(seq)** Converts a tuple into list.

Python includes following list methods

SN	Methods with Description
1	**list.append(obj)** Appends object obj to list
2	**list.count(obj)** Returns count of how many times obj occurs in list
3	**list. extend(seq)** Appends the contents of seq to list
4	**list.index(obj)** Returns the lowest index in list that obj appears
5	**list.insert(index, obj)** Inserts object obj into list at offset index
6	**list.pop(obj=list[-1])** Removes and returns last object or obj from list
7	**list.remove(obj)** Removes object obj from list
8	**list.reverse()** Reverses objects of list in place
9	**list.sort([func])** Sorts objects of list, use compare function if given

TUPLES

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

- **Accessing Values in Tuples:**

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0]

print "tup2[1:5]: ", tup2[1:5]
```

When the code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples:

Tuples are immutable which means you cannot update or change the values of tuple elements. We are able to take portions of existing tuples to create new tuples as the following example demonstrates −

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
tup3 = tup1 + tup2;
print tup3
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example:

```
tup = ('physics', 'chemistry', 1997, 2000);
print tup
del tup;
print "After deleting tup : "
print tup
```

Basic Tuples Operations:

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation

('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Built-in Tuple Functions

SN	Function with Description
1	**cmp(tuple1, tuple2)**:Compares elements of both tuples.
2	**len(tuple)**:Gives the total length of the tuple.
3	**max(tuple)**:Returns item from the tuple with max value.
4	**min(tuple)**:Returns item from the tuple with min value.
5	**tuple(seq)**:Converts a list into tuple.

3.2 DICTIONARY

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Accessing Values in Dictionary:

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}


print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Result –

```
dict['Name']:  Zara
dict['Age']:  7
```

Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}


dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

We can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example –

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}


del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary


print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Built-in Dictionary Functions & Methods –

Python includes the following dictionary functions −

SN	Function with Description
1	**cmp(dict1, dict2)** Compares elements of both dict.
2	**len(dict)** Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3	**str(dict)** Produces a printable string representation of a dictionary
4	**type(variable)** Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

SN	Methods with Description
1	**dict.clear()**:Removes all elements of dictionary *dict*
2	**dict. Copy()**:Returns a shallow copy of dictionary *dict*
3	**dict.fromkeys()**:Create a new dictionary with keys from seq and values *set* to *value*.
4	**dict.get(key, default=None)**:For *key* key, returns value or default if key not in dictionary
5	**dict.has_key(key)**:Returns *true* if key in dictionary *dict*, *false* otherwise
6	**dict.items()**:Returns a list of *dict*'s (key, value) tuple pairs
7	**dict.keys()**:Returns list of dictionary dict's keys
8	**dict.setdefault(key, default=None)**:Similar to get(), but will set dict[key]=default if *key* is not already in dict
9	**dict.update(dict2)**:Adds dictionary *dict2*'s key-values pairs to *dict*
10	**dict.values()**:Returns list of dictionary *dict*'s values

FUNCTIONS IN PYTHON

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

Defining a Function

Simple rules to define a function in Python.

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

```
# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;
# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

When the above code is executed, it produces the following result −

```
I'm first call to user defined function!
Again second call to the same function
```

Function Arguments

You can call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments
- ariable-length arguments

Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope. Following is a simple example −

```
total = 0; # This is global variable.
# Function definition is here
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2; # Here total is local variable.
   print "Inside the function local total : ", total
   return total;
sum( 10, 20 );
print "Outside the function global total : ", total
```

Result −

```
Inside the function local total :  30
Outside the function global total :  0
```

PYTHON MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
  print "Hello : ", par
  return
```

The *import* Statement

The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
def Pots():
  print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above −

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone,to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

```
# Now import your Phone Package.
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
```

RESULT:

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

PYTHON OBJECT ORIENTED

Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy. This chapter helps you become an expert in using Python's object-oriented programming support.

If you do not have any previous experience with object-oriented (OO) programming, you may want to consult an introductory course on it or at least a tutorial of some sort so that you have a grasp of the basic concepts.

However, here is small introduction of Object-Oriented Programming (OOP) to bring you at speed −

Overview of OOP Terminology

- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data member:** A class variable or instance variable that holds data associated with a class and its objects.
- **Function overloading:** The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or argument
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.
- **Inheritance:** The transfer of the characteristics of a class to other classes that are derived from it.

- **Instance:** An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
- **Instantiation:** The creation of an instance of a class.
- **Method :** A special kind of function that is defined in a class definition.
- **Object:** A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- **Operator overloading:** The assignment of more than one function to a particular operator.

Creating Classes

The *class* statement creates a new class definition. The name of the class immediately follows the keyword *class* followed by a colon as follows −

```
class ClassName:
   'Optional class documentation string'
   class_suite
```

- The class has a documentation string, which can be accessed via *ClassName.__doc__*.
- The *class_suite* consists of all the component statements defining class members, data attributes and functions.

Class Inheritance

Instead of starting from scratch, you can create a class by deriving it from a preexisting class by listing the parent class in parentheses after the new class name.

The child class inherits the attributes of its parent class, and you can use those attributes as if they were defined in the child class. A child class can also override data members and methods from the parent.

Syntax

Derived classes are declared much like their parent class; however, a list of base classes to inherit from is given after the class name −

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
   'Optional class documentation string'
   class_suite
```

Overriding Methods

You can always override your parent class methods. One reason for overriding parent's methods is because you may want special or different functionality in your subclass.

Example

```
class Parent:        # define parent class
   def myMethod(self):
      print 'Calling parent method'
class Child(Parent): # define child class
   def myMethod(self):
      print 'Calling child method'
c = Child()          # instance of child
c.myMethod()         # child calls overridden method
```

When the above code is executed, it produces the following result −

```
Calling child method
```

Base Overloading Methods

Following table lists some generic functionality that you can override in your own classes −

SN	Method, Description & Sample Call
1	**__init__ (self [,args...])** Constructor (with any optional arguments) Sample Call : *obj = className(args)*
2	**__del__(self)** Destructor, deletes an object Sample Call : *del obj*
3	**__repr__(self)** Evaluatable string representation Sample Call : *repr(obj)*
4	**__str__(self)** Printable string representation Sample Call : *str(obj)*
5	**__cmp__ (self, x)** Object comparison Sample Call : *cmp(obj, x)*

Overloading Operators

Suppose you have created a Vector class to represent two-dimensional vectors, what happens when you use the plus operator to add them? Most likely Python will yell at you.

You could, however, define the *__add__* method in your class to perform vector addition and then the plus operator would behave as per expectation −

Example

```
class Vector:
   def __init__(self, a, b):
      self.a = a
      self.b = b
   def __str__(self):
      return 'Vector (%d, %d)' % (self.a, self.b)
      def __add__(self,other):
      return Vector(self.a + other.a, self.b + other.b)
v1 = Vector(2,10)
v2 = Vector(5,-2)
print v1 + v2
```

When the above code is executed, it produces the following result −

```
Vector(7,8)
```

Data Hiding

An object's attributes may or may not be visible outside the class definition. You need to name attributes with a double underscore prefix, and those attributes then are not be directly visible to outsiders.

Example

```
class JustCounter:
   __secretCount = 0
  def count(self):
      self.__secretCount += 1
      print self.__secretCount
counter = JustCounter()
```

```
counter.count()

counter.count()

print counter.__secretCount
```

Result −

```
1

2

Traceback (most recent call last):

  File "test.py", line 12, in <module>

    print counter.__secretCount

AttributeError: JustCounter instance has no attribute '__secretCount'
```

Python protects those members by internally changing the name to include the class name. You can access such attributes as *object._className__attrName*. If you would replace your last line as following, then it works for you −

```
.....................

print counter._JustCounter__secretCount
```

When the above code is executed, it produces the following result −

```
1
2
2
```

Python MySQL Database Access

he Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

DESIGNING GUI APPLICATIONS USING PYQT IN PYTHON

- Difficulty Level : Medium
- Last Updated : 13 Oct, 2020

Building GUI applications using **PYQT designer tool** is comparatively less time consuming than code the widgets. It is one of the fastest and easiest ways to create GUIs. The normal approach is to write the code even for the widgets and for the functionalities as well. But using Qt-designer, one can simply drag and drop the widgets, which comes very useful while developing big-scale applications.

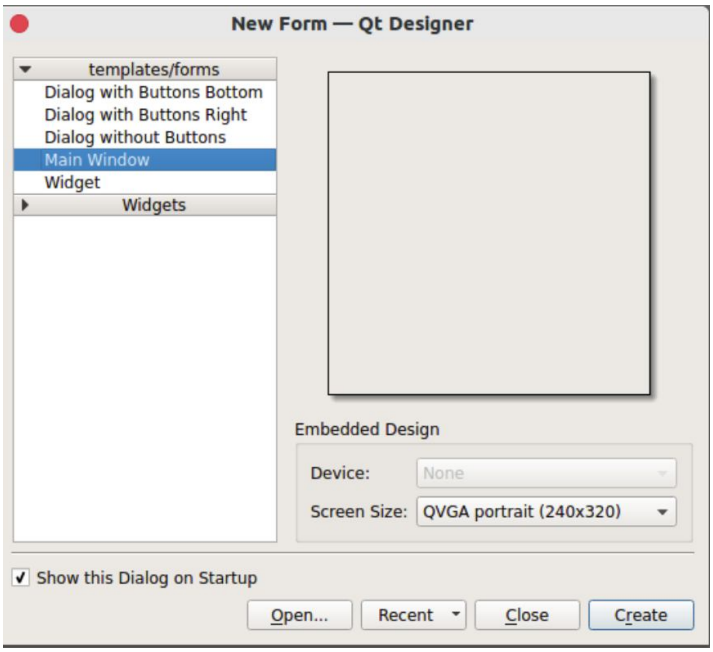
Installation of PyQt5 :

For Windows :

- pip install pyqt5
- pip install pyqt5-tools

Let's create a signup form using the QT designer tool. No code is required for creating forms, buttons, text boxes, etc! It is rather drag and drop environment. So, using PyQt is a lot simpler than Tkinter.

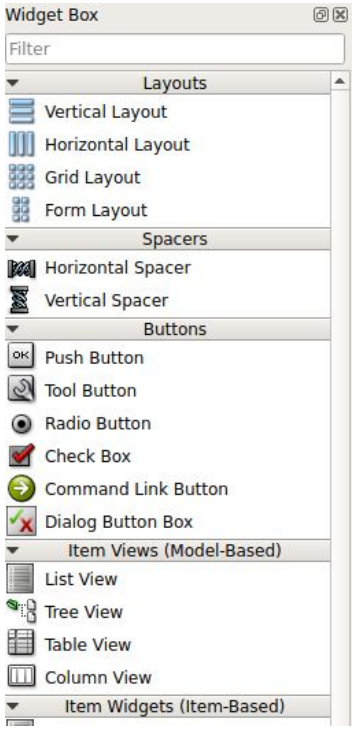
QT Designer will be located at MyPythonInstallationDir\Lib\site-packages\pyqt5-tools, and is named designer.exe (on Windows OS).
Open Qt Designer, then select **Main Window** and click **Create**. Set your preferred size of the window by dragging the edges of the window.



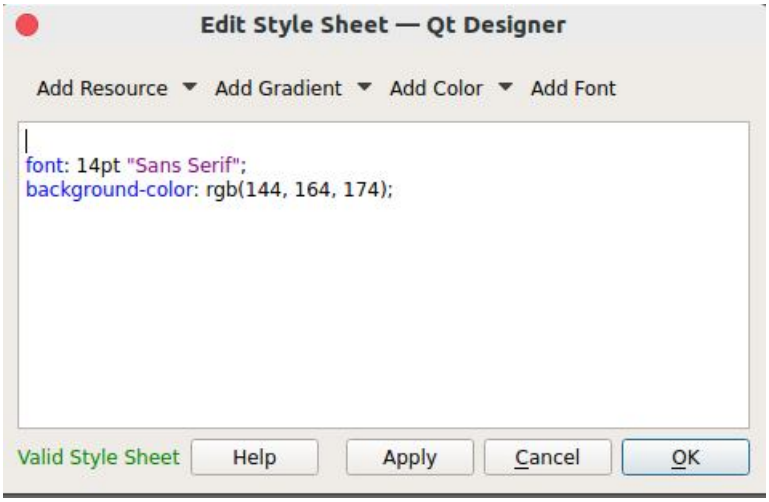
To create the layout of Singup form, following widgets are needed :

1. Three text edit boxes.
2. One button.
3. Four Text Labels (SignId Banner, UserName label, Password and Confirm Password label).

One has to find those widgets in **Widget Tool Box**. Just drag and drop the required widgets onto the Main Window or the window working on.



To change the appearance of the window or the widget, just right click on the widget and click **Change StyleSheet**.



To get preview of the window, press Ctrl + R .

Save the file :

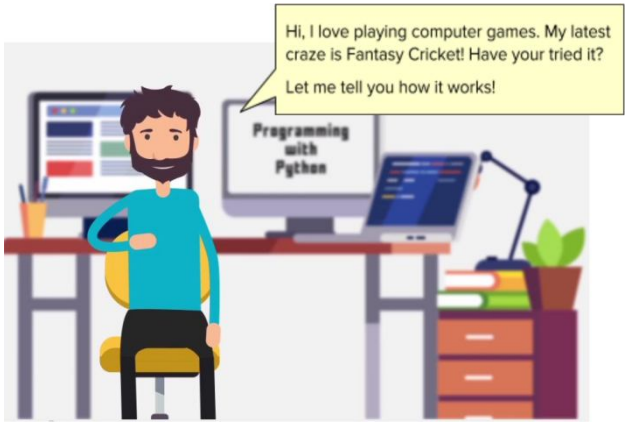
The file will be saved with **.ui** extension. To convert this file (**.ui** extension) to Python file (**.py** extension), follow these steps :

1. Open terminal and navigate to the folder where the layout.ui file is present.
2. To convert into Python file, type **pyuic5 -x layout.ui -o layout.py** on terminal.
3. Run layout.py using python!

Attention geek! Strengthen your foundations with the **Python Programming Foundation** Course and learn the basics.

To begin with, your interview preparations Enhance your Data Structures concepts with the **Python DS** Course.

PROJECT SCENARIO



Fantasy Cricket

It is an online game where you create a virtual team of real cricket players and score points depending on how your chosen players perform in real life matches. To win a tournament, you must try and get the maximum points and the No. 1 rank amongst other participants. Here's how a Fantasy Cricket game may look like.

Problem Statement

Create a Fantasy Cricket game in Python. The game should have all the features displayed in the mock-up screens in the scenario. To calculate the points for each player, you can use rules similar to the sample rules displayed below.

Sample of Rules

Batting

- 1 point for 2 runs scored
- Additional 5 points for half century
- Additional 10 points for century
- 2 points for strike rate (runs/balls faced) of 80-100
- Additional 4 points for strike rate>100
- 1 point for hitting a boundary (four) and 2 points for over boundary (six)

Bowling

- 10 points for each wicket
- Additional 5 points for three wickets per innings
- Additional 10 points for 5 wickets or more in innings
- 4 points for economy rate (runs given per over) between 3.5 and 4.5
- 7 points for economy rate between 2 and 3.5
- 10 points for economy rate less than 2

Fielding

- 10 points each for catch/stumping/run out

Database Design

For the database, you are required to use three tables – match, stats and teams.

match

Player	Scored	Faced	Fours	Sixes	Bowled	Maiden	Given	Wkts	Catches	Stumping	RO* *Run Out

teams

name	players	value

stats

player	matches	runs	100s	50s	value	ctg

Note: The teams table will be populated after score calculation. The data to enter in the remaining two tables is given below:

player	scored	faced	fours	sixes	bowled	maiden	given	wkts	catches	stumping	ro	value	matches	runs	100s	50s	ctg
Kohli	102	98	8	2	0	0	0	0	0	0	1	120	189	8257	28	43	BAT
Yuvraj	12	20	1	0	48	0	36	1	0	0	0	100	86	3589	10	21	BAT
Rahane	49	75	3	0	0	0	0	0	1	0	0	100	158	5435	11	31	BAT
Dhawan	32	35	4	0	0	0	0	0	0	0	0	85	25	565	2	1	AR
Dhoni	56	45	3	1	0	0	0	0	3	2	0	75	78	2573	3	19	BAT
Axar	8	4	2	0	48	2	35	1	0	0	0	100	67	208	0	0	BWL
Pandya	42	36	3	3	30	0	25	0	1	0	0	75	70	77	0	0	BWL
Jadeja	18	10	1	1	60	3	50	2	1	0	1	85	16	1	0	0	BWL
Kedar	65	60	7	0	24	0	24	0	0	0	0	90	111	675	0	1	BWL
Ashwin	23	42	3	0	60	2	45	6	0	0	0	100	136	1914	0	10	AR
Umesh	0	0	0	0	54	0	50	4	1	0	0	110	296	9496	10	64	WK
Bumrah	0	0	0	0	60	2	49	1	0	0	0	60	73	1365	0	8	WK
Bhuwaneshwar	15	12	2	0	60	1	46	2	0	0	0	75	17	289	0	2	AR
Rohit	46	65	5	1	0	0	0	0	1	0	0	85	304	8701	14	52	BAT
Kartick	29	42	3	0	0	0	0	0	2	0	1	75	11	111	0	0	AR

Hint: If you are wondering where to start and how to plan your work, here are some suggestions.

i. First, create the database of players. Plan the required tables and add data to your database.
ii. Next, create the GUI. Generate the required Python code for the UI.
iii. Finally, populate the Python code generated in step ii with more attributes and method definitions (action listeners).

CODING

MY PYTHON GAME MAIN CODE

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'main_code.ui'
#'M O T CODE'
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!

from calculate_points import player_points
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QMessageBox

from open import Ui_Dialog as Open   # importing open window dialogbox
from newteam import Ui_Dialog as New     # importing new window dialogbox
from evaluate import Ui_evaluate_team as Eva  # importing evaluate window dialogbox

import sqlite3
fant=sqlite3.connect('cricket_db.db')  # connecting to database file(fandatabase.db)
fantcurs=fant.cursor()


class Ui_MainWindow(object):
    def __init__(self):
        self.newDialog = QtWidgets.QMainWindow()
        self.new_screen = New()
        self.new_screen.setupUi(self.newDialog)

        self.EvaluateWindow = QtWidgets.QMainWindow()
        self.eval_screen = Eva()
        self.eval_screen.setupUi(self.EvaluateWindow)

        self.openDialog = QtWidgets.QMainWindow()
        self.open_screen = Open()
        self.open_screen.setupUi(self.openDialog)

      # FILE OPENING MENU
    def file_open(self):
        self.open_screen.setupUi(self.openDialog)
        self.openDialog.show()
        self.open_screen.openbtn.clicked.connect(self.openteam)

    # EVALUATE TEAM MENU
    def file_evaluate(self):
        self.eval_screen.setupUi(self.EvaluateWindow)
        self.EvaluateWindow.show()

    def setupUi(self, MainWindow):

        # INITIALISING POINTS AND COUNTS
        self.avail_points = 1000
        self.used_points = 0
        self.totalcount = 0
        self.batsmencount = 0
        self.bowlerscount = 0
        self.alrdscount = 0
        self.wicketerscount = 0
        # INITIALIZING LISTS
```

```
        self.a = []  # bowler names list
        self.b = []  #  batsman nameslist
        self.c = []   # allrounder names list


        self.d = []  #wicketer names list
        self.list1 = []    # selectedplayer's list


        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(556, 523)
        MainWindow.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.central_w = QtWidgets.QWidget(MainWindow)
        self.central_w.setObjectName("central_w")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.central_w)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Preferred)
        self.verticalLayout_3.addItem(spacerItem)
        self.frame = QtWidgets.QFrame(self.central_w)
        self.frame.setStyleSheet("background-color: rgb(241, 235, 255);\n"
"font: 75 10pt \"MS Shell Dlg 2\";\n"
"")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.gridLayout = QtWidgets.QGridLayout(self.frame)
        self.gridLayout.setContentsMargins(-1, 5, -1, 24)
        self.gridLayout.setObjectName("gridLayout")
        self.BAT = QtWidgets.QLabel(self.frame)
        self.BAT.setStyleSheet("color: rgb(0, 170, 255);")
        self.BAT.setObjectName("BAT")
        self.gridLayout.addWidget(self.BAT, 1, 1, 1, 1)
        self.label_2 = QtWidgets.QLabel(self.frame)
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 1, 0, 1, 1)
        self.WK = QtWidgets.QLabel(self.frame)
        self.WK.setStyleSheet("color: rgb(0, 170, 255);")
        self.WK.setObjectName("WK")
        self.gridLayout.addWidget(self.WK, 1, 7, 1, 1)
        self.label_4 = QtWidgets.QLabel(self.frame)
        self.label_4.setObjectName("label_4")
        self.gridLayout.addWidget(self.label_4, 1, 2, 1, 1)
        self.label_10 = QtWidgets.QLabel(self.frame)
        self.label_10.setObjectName("label_10")
        self.gridLayout.addWidget(self.label_10, 1, 4, 1, 1)
        self.ARL = QtWidgets.QLabel(self.frame)
        self.ARL.setStyleSheet("color: rgb(0, 170, 255);")
        self.ARL.setObjectName("ARL")
        self.gridLayout.addWidget(self.ARL, 1, 5, 1, 1)
        self.BOWL = QtWidgets.QLabel(self.frame)
        self.BOWL.setStyleSheet("color: rgb(0, 170, 255);")
        self.BOWL.setObjectName("BOWL")
        self.gridLayout.addWidget(self.BOWL, 1, 3, 1, 1)
        self.label_12 = QtWidgets.QLabel(self.frame)
        self.label_12.setObjectName("label_12")
        self.gridLayout.addWidget(self.label_12, 1, 6, 1, 1)
        self.label = QtWidgets.QLabel(self.frame)
        self.label.setStyleSheet("\n"
"font: italic 9pt \"Comic Sans MS\";")
```

```
    self.label.setObjectName("label")
        self.gridLayout.addWidget(self.label, 0, 0, 1, 1)
        self.verticalLayout_3.addWidget(self.frame)
        self.frame_2 = QtWidgets.QFrame(self.central_w)
        self.frame_2.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";\n"
"border-color: rgb(255, 255, 255);")

        self.frame_2.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.frame_2.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame_2.setLineWidth(0)
        self.frame_2.setObjectName("frame_2")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.frame_2)
        self.horizontalLayout.setContentsMargins(46, 8, 23, 5)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.label_6 = QtWidgets.QLabel(self.frame_2)
        self.label_6.setStyleSheet("font: 75 9pt \"MS Shell Dlg 2\";")
        self.label_6.setObjectName("label_6")
        self.horizontalLayout.addWidget(self.label_6)
        self.points_available = QtWidgets.QLabel(self.frame_2)
        self.points_available.setStyleSheet("color: rgb(85, 170, 255);")
        self.points_available.setObjectName("points_available")
        self.horizontalLayout.addWidget(self.points_available)
        self.label_7 = QtWidgets.QLabel(self.frame_2)
        self.label_7.setObjectName("label_7")
        self.horizontalLayout.addWidget(self.label_7, 0, QtCore.Qt.AlignRight)
        self.points_used = QtWidgets.QLabel(self.frame_2)
        self.points_used.setStyleSheet("color: rgb(85, 170, 255);")
        self.points_used.setObjectName("points_used")
        self.horizontalLayout.addWidget(self.points_used, 0, QtCore.Qt.AlignHCenter)
        self.verticalLayout_3.addWidget(self.frame_2)
        self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
        spacerItem1 = QtWidgets.QSpacerItem(13, 345, QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_4.addItem(spacerItem1)
        self.frame_3 = QtWidgets.QFrame(self.central_w)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.frame_3.sizePolicy().hasHeightForWidth())
        self.frame_3.setSizePolicy(sizePolicy)
        self.frame_3.setStyleSheet("border-color: rgb(15, 15, 15);")
        self.frame_3.setFrameShape(QtWidgets.QFrame.Box)
        self.frame_3.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.frame_3.setLineWidth(2)
        self.frame_3.setMidLineWidth(1)
        self.frame_3.setObjectName("frame_3")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.frame_3)
        self.verticalLayout.setContentsMargins(19, -1, 19, -1)
        self.verticalLayout.setObjectName("verticalLayout")
        self.frame_5 = QtWidgets.QFrame(self.frame_3)
        self.frame_5.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";")
        self.frame_5.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.frame_5.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.frame_5.setObjectName("frame_5")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.frame_5)
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
```

```
    self.bat_rb = QtWidgets.QRadioButton(self.frame_5) # bat_rb
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.bat_rb.sizePolicy().hasHeightForWidth())
        self.bat_rb.setSizePolicy(sizePolicy)
        self.bat_rb.setEnabled(False)
        self.bat_rb.setTabletTracking(False)
        self.bat_rb.setAcceptDrops(False)
        self.bat_rb.setObjectName("bat_rb")
        self.horizontalLayout_2.addWidget(self.bat_rb)


        self.bow_rb = QtWidgets.QRadioButton(self.frame_5)#bat_rb
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        self.bow_rb.setEnabled(False)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.bow_rb.sizePolicy().hasHeightForWidth())
        self.bow_rb.setSizePolicy(sizePolicy)
        self.bow_rb.setObjectName("bow_rb")

        self.horizontalLayout_2.addWidget(self.bow_rb)
        self.ar_rb = QtWidgets.QRadioButton(self.frame_5)#ar_rb
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.ar_rb.sizePolicy().hasHeightForWidth())
        self.ar_rb.setSizePolicy(sizePolicy)
        self.ar_rb.setEnabled(False)
        self.ar_rb.setObjectName("ar_rb")
        self.horizontalLayout_2.addWidget(self.ar_rb)


        self.wk_rb = QtWidgets.QRadioButton(self.frame_5)#wk_rb
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        self.wk_rb.setEnabled(False)
        sizePolicy.setHeightForWidth(self.wk_rb.sizePolicy().hasHeightForWidth())
        self.wk_rb.setSizePolicy(sizePolicy)
        self.wk_rb.setObjectName("wk_rb")
        self.horizontalLayout_2.addWidget(self.wk_rb)

        self.verticalLayout.addWidget(self.frame_5) #available_players
        self.av_player = QtWidgets.QListWidget(self.frame_3)
        self.av_player.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.av_player.setObjectName("av_player")
        self.verticalLayout.addWidget(self.av_player)

        self.horizontalLayout_4.addWidget(self.frame_3)
        spacerItem2 = QtWidgets.QSpacerItem(13, 345, QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_4.addItem(spacerItem2)
        self.label_9 = QtWidgets.QLabel(self.central_w)
        self.label_9.setStyleSheet("font: 20pt \"MS Shell Dlg 2\";")
        self.label_9.setLineWidth(-10)
        self.label_9.setMidLineWidth(-10)
        self.label_9.setObjectName("label_9")
        self.horizontalLayout_4.addWidget(self.label_9)
        spacerItem3 = QtWidgets.QSpacerItem(13, 345, QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Minimum)
```

```
        self.horizontalLayout_4.addItem(spacerItem3)
        self.frame_4 = QtWidgets.QFrame(self.central_w)
        self.frame_4.setFrameShape(QtWidgets.QFrame.Box)
        self.frame_4.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.frame_4.setLineWidth(2)
        self.frame_4.setMidLineWidth(1)
        self.frame_4.setObjectName("frame_4")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.frame_4)
        self.verticalLayout_2.setContentsMargins(18, 9, 14, -1)
        self.verticalLayout_2.setSpacing(6)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.frame_6 = QtWidgets.QFrame(self.frame_4)
        self.frame_6.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";")



        self.frame_6.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.frame_6.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_6.setObjectName("frame_6")
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout(self.frame_6)
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.label_8 = QtWidgets.QLabel(self.frame_6)
        self.label_8.setObjectName("label_8")
        self.horizontalLayout_3.addWidget(self.label_8, 0, QtCore.Qt.AlignHCenter)
        self.team_name = QtWidgets.QLabel(self.frame_6)
        self.team_name.setStyleSheet("color: rgb(85, 170, 255);")
        self.team_name.setObjectName("team_name")
        self.horizontalLayout_3.addWidget(self.team_name)

        self.verticalLayout_2.addWidget(self.frame_6) #select_players
        self.sel_player = QtWidgets.QListWidget(self.frame_4)
        self.sel_player.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.sel_player.setObjectName("sel_player")
        self.verticalLayout_2.addWidget(self.sel_player)
        self.horizontalLayout_4.addWidget(self.frame_4)
        spacerItem4 = QtWidgets.QSpacerItem(13, 345, QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_4.addItem(spacerItem4)
        self.verticalLayout_3.addLayout(self.horizontalLayout_4)
        MainWindow.setCentralWidget(self.central_w)


        self.menubar = QtWidgets.QMenuBar(MainWindow) #Menubar
        self.menubar.setGeometry(QtCore.QRect(0, 0, 556, 21))
        self.menubar.setObjectName("menubar")
        self.menuManage_Teams = QtWidgets.QMenu(self.menubar)
        self.menuManage_Teams.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.menuManage_Teams.setStyleSheet("color: rgb(0, 170, 255);")
        self.menuManage_Teams.setStyleSheet("background-color: rgb(232, 232, 232);")
        self.menuManage_Teams.setStyleSheet("selection-background-color: rgb(0, 170, 255);\n""\n""")
        self.menuManage_Teams.setObjectName("menuManage_Teams")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.new_t = QtWidgets.QAction(MainWindow)#new window
        self.new_t.setIconText("NEW Team")
        self.new_t.setShortcut("Ctrl+N")
```

```
 self.new_t.setShortcutContext(QtCore.Qt.WindowShortcut)
    self.new_t.setVisible(True)
    self.new_t.setShortcutVisibleInContextMenu(True)
    self.new_t.setObjectName("new_t")

    self.open_t = QtWidgets.QAction(MainWindow)#open team
    self.open_t.setObjectName("open_t")

    self.save_t = QtWidgets.QAction(MainWindow)#save team
    self.save_t.setObjectName("save_t")

    self.evaluate_t = QtWidgets.QAction(MainWindow)#EVALUATETEAM
    self.evaluate_t.setObjectName("evaluate_t")

    self.actionQuit = QtWidgets.QAction(MainWindow)# Quitwindow
    self.actionQuit.setObjectName("actionQuit")
    self.actionQuit.triggered.connect(self.quit)



    self.menuManage_Teams.addAction(self.new_t)
    self.menuManage_Teams.addAction(self.open_t)
    self.menuManage_Teams.addAction(self.save_t)
    self.menuManage_Teams.addAction(self.evaluate_t)
    self.menuManage_Teams.addAction(self.actionQuit)
    self.menubar.addAction(self.menuManage_Teams.menuAction())

    #Action teiggered
    self.new_t.triggered.connect(self.file_new)
    self.open_t.triggered.connect(self.file_open)
    self.save_t.triggered.connect(self.file_save)
    self.evaluate_t.triggered.connect(self.file_evaluate)
    self.actionQuit.triggered.connect(self.quit)

    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)
    # DOUBLE CLICK
    self.av_player.itemDoubleClicked.connect(self.removelist1)
    self.sel_player.itemDoubleClicked.connect(self.removelist2)

    # -----stats of player
    self.stats = {}

    self.new_screen.savename.clicked.connect(self.namechange)

    # RADIOBUTTONS  CLICK
    self.bat_rb.clicked.connect(self.load_names)
    self.wk_rb.clicked.connect(self.load_names)
    self.bow_rb.clicked.connect(self.load_names)
    self.ar_rb.clicked.connect(self.load_names)


def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
    self.BAT.setText(_translate("MainWindow", "##"))
    self.label_2.setText(_translate("MainWindow", "Batsman (BAT)"))
    self.WK.setText(_translate("MainWindow", "##"))
    self.label_4.setText(_translate("MainWindow", "Bowlers (BOW)"))
    self.label_10.setText(_translate("MainWindow", "AllRounders (AR) "))
```

```
        self.ARL.setText(_translate("MainWindow", "##"))
        self.BOWL.setText(_translate("MainWindow", "##"))
        self.label_12.setText(_translate("MainWindow", "Wicket-Keeper (WK)"))
        self.label.setText(_translate("MainWindow", "Your Selection"))
        self.label_6.setText(_translate("MainWindow", "Points Availables"))
        self.points_available.setText(_translate("MainWindow", "####"))
        self.label_7.setText(_translate("MainWindow", "Points Used"))
        self.points_used.setText(_translate("MainWindow", "####"))
        self.bat_rb.setText(_translate("MainWindow", "BAT"))
        self.bow_rb.setText(_translate("MainWindow", "BOW"))
        self.ar_rb.setText(_translate("MainWindow", "AR"))
        self.wk_rb.setText(_translate("MainWindow", "WK"))
        self.label_9.setText(_translate("MainWindow", ">"))
        self.label_8.setText(_translate("MainWindow", "Team Name  :"))
        self.team_name.setText(_translate("MainWindow", "Displayed Here"))
        self.menuManage_Teams.setTitle(_translate("MainWindow", "Manage Teams"))
        self.new_t.setText(_translate("MainWindow", "NEW Team"))
        self.new_t.setShortcut(_translate("MainWindow", "Ctrl+N"))
        self.open_t.setText(_translate("MainWindow", "OPEN Team"))
        self.open_t.setShortcut(_translate("MainWindow", "Ctrl+O"))
        self.save_t.setText(_translate("MainWindow", "SAVE team"))




        self.save_t.setShortcut(_translate("MainWindow", "Ctrl+S"))
        self.evaluate_t.setText(_translate("MainWindow", "EVALUATE Team"))
        self.evaluate_t.setShortcut(_translate("MainWindow", "Ctrl+E"))
        self.actionQuit.setText(_translate("MainWindow", "QUIT"))
    # NEW FILE MENU
    def file_new(self):
        self.newDialog.show()

    def namechange(self):
        teamname = self.new_screen.team_name.text()
        fantcurs.execute("SELECT DISTINCT name FROM teams")
        l = fantcurs.fetchall()
        for i in l:
            if i[0] == teamname:
                msg = QtWidgets.QMessageBox()
                msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Team with same name already exists!!\nPlease choose another name")
                msg.setWindowTitle("Invalid Team Name")
                msg.exec_()
                return 0
        if len(teamname) == 0:
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Warning)
            msg.setText("You cannot leave the field blank!!!")
            msg.setWindowTitle("Invalid Team Name")
            msg.exec_()
            return 0
        elif teamname.isnumeric():
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Warning)
            msg.setText("Please enter a valid teamname\n(Name must contain atleast one character)!!")
            msg.setWindowTitle("Invalid Team Name")
            msg.exec_()
            return 0
        else:
```

```
            self.reset()
            self.tname = self.new_screen.team_name.text()
            self.team_name.setText(str('    '+self.tname))
            self.newDialog.close()

    #TO RESET ALL COUNTS AND LITS
    def reset(self):
        self.enablebuttons()
        self.load_names()
        self.used_points = 0
        self.alrdscount = 0
        self.wicketerscount = 0
        self.batsmencount = 0
        self.bowlerscount = 0
        self.totalcount = 0
        self.avail_points = 1000
        self.points_available.setText(str(self.avail_points))
        self.points_used.setText(str(self.used_points))
        self.BOWL.setText(str(self.bowlerscount))
        self.BAT.setText(str(self.batsmencount))
        self.ARL.setText(str(self.alrdscount))
        self.WK.setText(str(self.wicketerscount))
        self.list1.clear()
        self.load_names()

        self.sel_player.clear()


        #SAVE TEAM MENU
    def file_save(self):
        if not self.error():  #IF THERE IS AN ERROR
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setInformativeText(' Inufficient Players OR Points !!')
            msg.setWindowTitle("Fantasy Cricket")
            msg.exec_()
        elif self.error():  # IF NO ERROR
            try:
                fantcurs.execute("SELECT DISTINCT name FROM teams;")
                x = fantcurs.fetchall()
                for i in x:
                    if self.team_name.text() == i[0]:   # CHECKING IF THE TEAMNAME ALREADY EXISTS
                        print('Updating already there')
                        fantcurs.execute("DELETE  FROM teams WHERE name='" + self.team_name.text() + "';")
#DELETING TO UPDATE TEAM
            except:
                print('error')
            for i in range(self.sel_player.count()):
                # print('----addding--')
                # print('teamnane: ',self.team_name.text())
                # print('playername: ',self.list1[i])
                # print('points: ', player_points[self.list1[i]])
                try:
                    fantcurs.execute("INSERT INTO teams (name,players,value) VALUES (?,?,?)",
                                     (self.team_name.text(), self.list1[i], player_points[self.list1[i]]))
```

```
                # self.file_evaluate()
            except:
                print('error in operation!')
        fant.commit()
    else:
        print('---error in operation')


 # QUITING METHOD
 def quit(self):
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Critical)
    msg.setInformativeText(' Bye Bye')
    msg.setWindowTitle("Fantasy Cricket")
    msg.exec_()
    # print('exit')
    sys.exit()

#ON RADIOBUTTONS CLICKED
 def load_names(self):
    Batsman = 'BAT'
    WicketKeeper = 'WK'
    Allrounder = 'AR'
    Bowler = 'BWL'
    sql1 = "SELECT player,value from stats WHERE ctg = '" + Batsman + "';"
    sql2 = "SELECT Player,value from stats WHERE ctg = '" + WicketKeeper + "';"
    sql3 = "SELECT Player,value from stats WHERE ctg ='" + Allrounder + "';"
    sql4 = "SELECT Player,value from stats WHERE ctg = '" + Bowler + "';"




    fantcurs.execute(sql1)
    x = fantcurs.fetchall()
    fantcurs.execute(sql4)
    y = fantcurs.fetchall()
    fantcurs.execute(sql3)
    z = fantcurs.fetchall()
    fantcurs.execute(sql2)
    w = fantcurs.fetchall()

    batsmen = []
    bowlers = []
    allrounders = []
    wcktkeepers = []

    for k in x:
        batsmen.append(k[0])
        self.b.append(k[0])
        self.stats[k[0]] = k[1]
    for k in y:
        bowlers.append(k[0])
        self.stats[k[0]] = k[1]
        self.a.append(k[0])
    for k in w:
        wcktkeepers.append(k[0])
        self.stats[k[0]] = k[1]
        self.d.append(k[0])
    for k in z:
```

```
            allrounders.append(k[0])
            self.stats[k[0]] = k[1]
            self.c.append(k[0])
        for i in self.list1:
            if i in allrounders:
                allrounders.remove(i)
            elif i in batsmen:
                batsmen.remove(i)
            elif i in bowlers:
                bowlers.remove(i)
            elif i in wcktkeepers:
                wcktkeepers.remove(i)

        if self.bat_rb.isChecked() == True:
            self.av_player.clear()
            for i in range(len(batsmen)):
                item = QtWidgets.QListWidgetItem(batsmen[i])
                font = QtGui.QFont()
                font.setBold(True)
                font.setWeight(75)
                item.setFont(font)
                self.av_player.addItem(item)
        elif self.bow_rb.isChecked() == True:
            self.av_player.clear()
            for i in range(len(bowlers)):
                item = QtWidgets.QListWidgetItem(bowlers[i])
                font = QtGui.QFont()
                font.setBold(True)
                font.setWeight(75)
                item.setFont(font)
                self.av_player.addItem(item)
        elif self.ar_rb.isChecked() == True:
            self.av_player.clear()
            for i in range(len(allrounders)):




                item = QtWidgets.QListWidgetItem(allrounders[i])
                font = QtGui.QFont()
                font.setBold(True)
                font.setWeight(75)
                item.setFont(font)
                self.av_player.addItem(item)

        elif self.wk_rb.isChecked() == True:
            self.av_player.clear()
            for i in range(len(wcktkeepers)):
                item = QtWidgets.QListWidgetItem(wcktkeepers[i])
                font = QtGui.QFont()
                font.setBold(True)
                font.setWeight(75)
                item.setFont(font)
                self.av_player.addItem(item)

    def removelist1(self, item):   # REMOVE FROM AVAILABLE PLAYERS AND ADD TO SELECTED
PLAYERS
        self.conditions_1(item.text())
        self.av_player.takeItem(self.av_player.row(item))
```

```
        self.sel_player.addItem(item.text())
        self.totalcount = self.sel_player.count()
        self.list1.append(item.text())
        self.error()

    def conditions_1(self, cat):   # Adding and Deducting respective points from points_calculator.py
        self.avail_points -= self.stats[cat]
        self.used_points += self.stats[cat]
        if cat in self.a:
            self.bowlerscount += 1
        elif cat in self.d:
            self.wicketerscount += 1
        elif cat in self.c:
            self.alrdscount += 1
        elif cat in self.b:
            self.batsmencount += 1

        self.points_available.setText(str(self.avail_points))
        self.points_used.setText(str(self.used_points))
        self.BOWL.setText(str(self.bowlerscount))
        self.BAT.setText(str(self.batsmencount))
        self.ARL.setText(str(self.alrdscount))
        self.WK.setText(str(self.wicketerscount))

    def conditions_2(self, cat):   # Adding and Deducting respective poinrs from points_calculator.py
        self.avail_points += self.stats[cat]
        self.used_points -= self.stats[cat]
        if cat in self.a:
            self.bowlerscount -= 1
        elif cat in self.d:
            self.wicketerscount -= 1
        elif cat in self.c:
            self.alrdscount -= 1
        elif cat in self.b:
            self.batsmencount -= 1

        self.points_available.setText(str(self.avail_points))
        self.points_used.setText(str(self.used_points))
        self.BOWL.setText(str(self.bowlerscount))
        self.BAT.setText(str(self.batsmencount))




        self.ARL.setText(str(self.alrdscount))
        self.WK.setText(str(self.wicketerscount))

    def removelist2(self, item):   # REMOVE FROM SELECTED PLAYERS AND ADD TO AVAIALBLE
PLAYERS
        self.sel_player.takeItem(self.sel_player.row(item))
        self.av_player.addItem(item.text())
        self.list1.remove(item.text())
        # self.error()
        self.totalcount = self.sel_player.count()
        self.conditions_2(item.text())

    def openteam(self):  #upon open team selected
        self.reset()
        teamname = self.open_screen.open_cb.currentText()
        self.team_name.setText(teamname)
        self.enablebuttons()
```

```
        fantcurs.execute("SELECT players from teams WHERE name= '" + teamname + "';")
        x=fantcurs.fetchall()
        score=[]
        for i in x:
            fantcurs.execute("SELECT value from stats WHERE player='"+i[0]+"';")
            y=fantcurs.fetchone()
            score.append(y[0])
        # print(score)
        sum=0
        for i in score:
            sum+=i
        self.sel_player.clear()
        self.load_names()
        for i in x:
            self.sel_player.addItem(i[0])
            self.list1.append(i[0])
            self.conditions_1(i[0])
        self.used_points = sum
        self.avail_points = 1000 - sum
        self.points_available.setText(str(self.avail_points))
        self.points_used.setText(str(self.used_points))
        self.openDialog.close()

    def enablebuttons(self):
        self.bat_rb.setEnabled(True)
        self.bow_rb.setEnabled(True)
        self.ar_rb.setEnabled(True)
        self.wk_rb.setEnabled(True)

    def disablebuttons(self):
        self.bat_rb.setEnabled(False)
        self.bow_rb.setEnabled(False)
        self.ar_rb.setEnabled(False)
        self.wk_rb.setEnabled(False)


    def error(self):
        msg = QMessageBox()
        if self.wicketerscount > 1:
            msg.setIcon(QMessageBox.Critical)
            # msg.setText("Error")
            msg.setInformativeText('Only 1 wicketkeeper is allowed!')
            msg.setWindowTitle("Error")
            msg.exec_()
            return 0



        elif self.totalcount > 11:
            msg.setIcon(QMessageBox.Critical)
            msg.setInformativeText('No more than 11 players allowed!')
            msg.setWindowTitle("Selection Error")
            msg.exec_()
            return 0
        elif self.totalcount < 11 :
            return 0
        elif self.wicketerscount < 1:
            return 0
        elif self.avail_points <= -1:
            msg.setIcon(QMessageBox.Critical)
            msg.setInformativeText('Not enough points!')
```

```
            msg.setWindowTitle("Selection Cricket")
            msg.exec_()
            return 0

        return 1



if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Calculate Points

```
import sqlite3
#connecting to cricket_db.db
db = sqlite3.connect("cricket_db.db")
cursor = db.cursor()
cursor.execute("SELECT * FROM match")
row = cursor.fetchall()


def calculate_points(row):
    points = 0.0
    score = row[1]
    try:
        strike_rate = float(row[1]) / float(row[2])  # strike rate =runs/balls faced
    except:
        strike_rate = 0
    fours, sixes = float(row[3]), float(row[4])


    twos = int((score - 4 * fours - 6 * sixes) / 2)
    wickets = 10 * float(row[8])
    try:
        economy = float(row[7]) / (float(row[5]) / 6)
    except:
        economy = 0
    Fielding = float(row[9]) + float(row[10]) + float(row[11])
```

```
    # 1 point for hitting a boundary and 2 runs ,2 points for over boundary,10 points each for catch and wicket
    points += (fours + 2 * sixes + 10 * Fielding + twos + wickets)
    if score > 100:
        points += 10  # 10 points for century
    elif score >= 50:
        points += 5  # 5 points for half century
    if strike_rate > 1:  # for strike rate>100
        points += 4


    elif strike_rate >= 0.8:
        points += 2  # 2 points for strike rate >= 80
    if wickets >= 5:
        points += 10  # Additional 10 points for 5 wickets
    elif wickets > 3:
        points += 5  # Additional 5 points for 3 wickets
    if economy >= 3.5 and economy <= 4.5:
        points += 4  # 4 points for eco rate between 3.5 and 4.5
    elif economy >= 2 and economy < 3.5:
        points += 7  # 7 points for economy rate between 2 and 3.5
    elif economy < 2:
        points += 10  # 10 points for economy rate less than 2
    return points


player_points = {}
for i in row:
    player_points[i[0]] = calculate_points(i)


print(player_points)
```

cricket_db

```
import sqlite3


# connecting to database file
mycurs = sqlite3.connect('cricket_db.db')
curs = mycurs.cursor()

#CREATING MATCH TABLE
curs.execute('''CREATE TABLE IF NOT EXISTS match (player TEXT NOT
NULL,scored INTEGER,faced INTEGER,fours INTEGER,sixes INTEGER,bowled
INTEGER,maiden INTEGER,given INTEGER,wkts INTEGER,catches
INTEGER,stumping INTEGER,ro INTEGER);''')



#CREATING STATS TABLE
curs.execute('''CREATE TABLE IF NOT EXISTS stats (player PRIMARY
KEY,matches INTEGER,runs INTEGER,hundreds INTEGER,fifties INTEGER,value
INTEGER,ctg TEXT NOT NULL);''')



#CREATING TEAMS TABLE
curs.execute('''CREATE TABLE IF NOT EXISTS teams (name TEXT NOT
NULL,players TEXT NOT NULL,value INTEGER);''')



#DISPLAY DATA IF EXISTS IN DATABASE
sql="select * from match"
curs.execute(sql)
result=curs.fetchall()
if(result):
   for i in result:


      print(i)
   opt=input("\n add more players details ? (Y/N) : ")
else:
   print("No any players data found ")

   opt=input("\n add players data (Y/N) :")
#ADDING DATA FROM USER TO MATCH TABLE
while(opt=='y' or opt=='Y'):
```

```
    row=[input("Player name :")]
    row.append(int(input("Score:")))
    row.append(int(input("Faced: ")))
    row.append(int(input("Fours: ")))
    row.append(int(input("Sixes: ")))
    row.append(int(input("Bowled: ")))
    row.append(int(input("Maiden: ")))
    row.append(int(input("Given: ")))
    row.append(int(input("Wkts: ")))
    row.append(int(input("Catches: ")))
    row.append(int(input("Stumping: ")))
    row.append(int(input("RO: ")))
        try:
        curs.execute("INSERT INTO match (player,scored, faced,
fours,sixes,bowled,maiden,given,wkts,catches,stumping,ro) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?)", #adding data to database
                    (row[0],row[1], row[2],
row[3],row[4],row[5],row[6],row[7],row[8],row[9],row[10],row[11]))
        mycurs.commit()

        print("records added successfully match table.")
    except:    # except block to handle exceptions
        print("Error in operation.")
        mycurs.rollback()
    #ADDING DATA TO STATS TABLE FROM USER
    print("player information for State table ")
    row.append(int(input("Total matches: ")))
    row.append(int(input("Total runs: ")))
    row.append(int(input("100s: ")))
    row.append(int(input("50s: ")))
    row.append(int(input("Value: ")))
    row.append(input("Category as (BAT,BWL,AR,WK): "))

    try:       #try block to catch exceptions

        curs.execute("INSERT INTO stats (player,matches,runs, hundreds,
fifties,value,ctg) VALUES (?,?,?,?,?,?,?)", #adding data to database
                    (row[0],row[12], row[13], row[14],row[15],row[16],row[17]))
        mycurs.commit()

        print("records added successfully for stats table.")
    except:    # except block to handle exceptions
        print("Error in operation.")
        mycurs.rollback()
```

```
    opt=input("adding more player ? (Y/N) : ")


print("bye")
curs.close() #close database
```

Evaluate

```
  # -*- coding: utf-8 -*-


# Form implementation generated from reading ui file 'evaluate.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
from scoreboard import Ui_Dialog as Score  #Score window
import sqlite3
match= sqlite3.connect("cricket_db.db")
matchcur=match.cursor()


class Ui_evaluate_team(object):
    def __init__(self):  #initialising score window
        self.scoreDialog = QtWidgets.QMainWindow()
        self.score_screen = Score()
        self.score_screen.setupUi(self.scoreDialog)


    def setupUi(self, evaluate_team):
        evaluate_team.setObjectName("evaluate_team")
        evaluate_team.resize(634, 504)
        evaluate_team.setStyleSheet("background-color: rgb(199, 199, 199);")
        self.label = QtWidgets.QLabel(evaluate_team)
        self.label.setGeometry(QtCore.QRect(90, 20, 391, 51))
        font = QtGui.QFont()
        font.setFamily("Comic Sans MS")
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.selectteam_cb = QtWidgets.QComboBox(evaluate_team)
        self.selectteam_cb.setGeometry(QtCore.QRect(90, 70, 141, 21))
```

```
        self.selectteam_cb.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"font: 75 italic 10pt \"Century Gothic\";")

        self.selectteam_cb.setObjectName("selectteam_cb")
        self.selectteam_cb.addItem("")
        self.selectmatch_cb = QtWidgets.QComboBox(evaluate_team)
        self.selectmatch_cb.setEnabled(True)
        self.selectmatch_cb.setGeometry(QtCore.QRect(360, 70, 141, 22))
        font = QtGui.QFont()
        font.setFamily("Century Gothic")
        font.setPointSize(11)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(9)
        self.selectmatch_cb.setFont(font)
        self.selectmatch_cb.setStyleSheet("background-color: rgb(255, 255, 255);\n"
"font: 75 italic 11pt \"Century Gothic\";")
        self.selectmatch_cb.setObjectName("selectmatch_cb")
        self.selectmatch_cb.addItem("")
        self.selectmatch_cb.addItem("")
        self.line = QtWidgets.QFrame(evaluate_team)
        self.line.setGeometry(QtCore.QRect(40, 95, 551, 41))
        self.line.setStyleSheet("color: rgb(0, 0, 0);")
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.label_2 = QtWidgets.QLabel(evaluate_team)
        self.label_2.setGeometry(QtCore.QRect(80, 160, 81, 21))
        font = QtGui.QFont()
        font.setFamily("Comic Sans MS")
        font.setPointSize(12)
        font.setBold(False)
        font.setWeight(50)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(evaluate_team)
        self.label_3.setGeometry(QtCore.QRect(350, 150, 81, 21))
        font = QtGui.QFont()
        font.setFamily("Comic Sans MS")
        font.setPointSize(12)
        font.setBold(False)
        font.setWeight(50)
```

```
self.label_3.setFont(font)
self.label_3.setObjectName("label_3")
self.calcscore_btn = QtWidgets.QPushButton(evaluate_team)
self.calcscore_btn.setEnabled(True)



self.calcscore_btn.setGeometry(QtCore.QRect(240, 410, 141, 31))
font = QtGui.QFont()
font.setFamily("Comic Sans MS")
font.setPointSize(10)
self.calcscore_btn.setFont(font)
self.calcscore_btn.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
self.calcscore_btn.setAutoDefault(False)
self.calcscore_btn.setObjectName("calcscore_btn")
self.players_lw = QtWidgets.QListWidget(evaluate_team)
self.players_lw.setGeometry(QtCore.QRect(80, 180, 191, 221))
font = QtGui.QFont()
font.setFamily("Comic Sans MS")
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.players_lw.setFont(font)
self.players_lw.setStyleSheet("background-color: rgb(255, 255, 255);")
self.players_lw.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
self.players_lw.setDefaultDropAction(QtCore.Qt.IgnoreAction)
self.players_lw.setSelectionMode(QtWidgets.QAbstractItemView.NoSelection)
self.players_lw.setObjectName("players_lw")
self.scores_lw = QtWidgets.QListWidget(evaluate_team)
self.scores_lw.setGeometry(QtCore.QRect(340, 180, 191, 221))
font = QtGui.QFont()
font.setFamily("Comic Sans MS")
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.scores_lw.setFont(font)
self.scores_lw.setStyleSheet("background-color: rgb(255, 255, 255);")
self.scores_lw.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
self.scores_lw.setSelectionMode(QtWidgets.QAbstractItemView.NoSelection)
self.scores_lw.setObjectName("scores_lw")

self.retranslateUi(evaluate_team)
QtCore.QMetaObject.connectSlotsByName(evaluate_team)
```

```
        self.calcscore_btn.clicked.connect(self.final_score)
        selected_team = self.selectteam_cb.currentText()

        self.changedname(selected_team)
        self.selectteam_cb.currentTextChanged.connect(self.changedname)

    def retranslateUi(self, evaluate_team):
        _translate = QtCore.QCoreApplication.translate
        evaluate_team.setWindowTitle(_translate("evaluate_team", "Dialog"))
        self.label.setText(_translate("evaluate_team", "Evaluate the Performance of your
Fantasy Team"))
        self.selectteam_cb.setCurrentText(_translate("evaluate_team", "Select team"))
        self.selectteam_cb.setItemText(0, _translate("evaluate_team", "Select team"))
        self.selectmatch_cb.setItemText(0, _translate("evaluate_team", "select match"))
        self.selectmatch_cb.setItemText(1, _translate("evaluate_team", "Match1"))
        self.label_2.setText(_translate("evaluate_team", "Players"))
        self.label_3.setText(_translate("evaluate_team", "Points"))
        self.calcscore_btn.setStatusTip(_translate("evaluate_team", "calculating score"))
        self.calcscore_btn.setText(_translate("evaluate_team", "Calculate Score"))
        x = matchcur.execute("SELECT  DISTINCT name from teams;")
        team = x.fetchall()
        for i in team:
            self.selectteam_cb.addItem(i[0])
    def changedname(self, t):
        self.players_lw.clear()
        self.scores_lw.clear()
        y = matchcur.execute("SELECT players from teams WHERE name='" + t + "';")
        player = y.fetchall()

        # print('player',player)
        for j in player:
            self.players_lw.addItem(j[0])
        z = matchcur.execute("SELECT value from teams WHERE name='" + t + "';")
        value = z.fetchall()
        for k in value:
            self.scores_lw.addItem(str(k[0]))
```

```
    def final_score(self):
        total_score=0
        t=self.selectteam_cb.currentText()   # current teamname
        # print(t)
        z = matchcur.execute("SELECT value from teams WHERE name='" + t + "';")
        value = z.fetchall()

        # print('value', value)
        for k in value:
            total_score+=k[0]
        self.score_screen.finalscore.setText(str(total_score))  # opening score dialog box
and setting final score
        self.scoreDialog.show()


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    evaluate_team = QtWidgets.QDialog()
    ui = Ui_evaluate_team()
    ui.setupUi(evaluate_team)
    evaluate_team.show()
    sys.exit(app.exec_())
```

New Team

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'newteam.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(407, 260)
        self.frame = QtWidgets.QFrame(Dialog)
        self.frame.setGeometry(QtCore.QRect(0, -20, 401, 231))
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.label = QtWidgets.QLabel(self.frame)
        self.label.setGeometry(QtCore.QRect(110, 50, 210, 22))
        font = QtGui.QFont()
        font.setFamily("Segoe UI Semibold")
        font.setPointSize(11)
        font.setBold(True)
        font.setItalic(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.team_name = QtWidgets.QLineEdit(self.frame)
        self.team_name.setGeometry(QtCore.QRect(90, 91, 221, 41))
        self.team_name.setObjectName("team_name")



        self.savename = QtWidgets.QPushButton(self.frame)
        self.savename.setGeometry(QtCore.QRect(140, 160, 93, 28))
        font = QtGui.QFont()
        font.setFamily("MS Sans Serif")
        font.setPointSize(8)
        font.setBold(False)
        font.setItalic(True)
        font.setWeight(50)
        self.savename.setFont(font)
        self.savename.setStyleSheet("font: italic 8pt \"MS Sans Serif\";")
        self.savename.setObjectName("savename")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
```

```
    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "new_team"))
        self.label.setText(_translate("Dialog", "Create New Team"))
        self.team_name.setPlaceholderText(_translate("Dialog", "enter team name"))
        self.savename.setText(_translate("Dialog", "Save"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

Open

```
 # -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'open.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
import sqlite3
match=sqlite3.connect('cricket_db.db')
matchcur=match.cursor()

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(368, 274)
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(80, 30, 299, 40))
        font = QtGui.QFont()
        font.setFamily("Segoe Print")
        font.setPointSize(14)
```

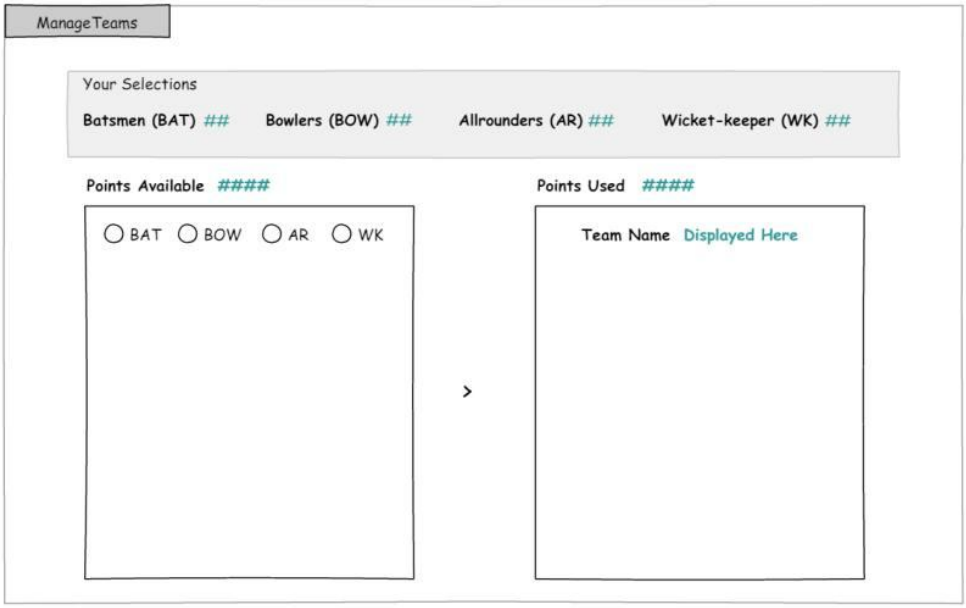
```
        font.setBold(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.openbtn = QtWidgets.QPushButton(Dialog)
        self.openbtn.setGeometry(QtCore.QRect(120, 160, 93, 49))
        font = QtGui.QFont()
        font.setFamily("Microsoft Sans Serif")
        font.setPointSize(10)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)
        self.openbtn.setFont(font)
        self.openbtn.setStyleSheet("font: 10pt \"Microsoft Sans Serif\";")
        self.openbtn.setObjectName("openbtn")
        self.open_cb = QtWidgets.QComboBox(Dialog)
        self.open_cb.setGeometry(QtCore.QRect(70, 100, 211, 31))
        self.open_cb.setObjectName("open_cb")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

        teams= matchcur.execute("SELECT DISTINCT name FROM teams;")  # fetching
team names
        y= teams.fetchall()
        for i in y:
            self.open_cb.addItem(i[0])



            def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.label.setText(_translate("Dialog", "select team  to open"))
        self.openbtn.setText(_translate("Dialog", "open"))



if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

scoreboard

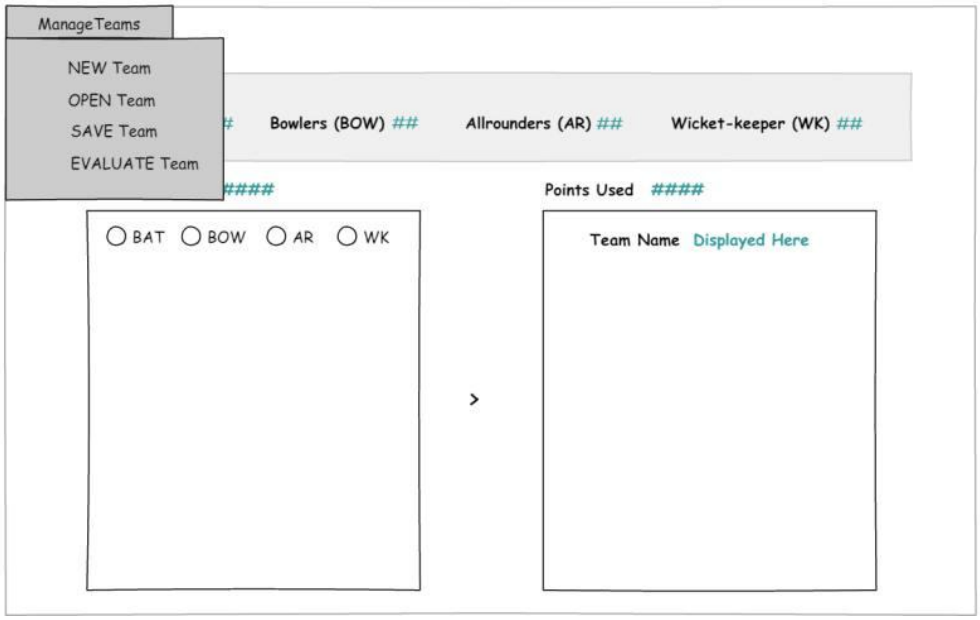
```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'scoreboard.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(360, 244)
        Dialog.setStyleSheet("background-color: rgb(235, 235, 235);")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(80, 60, 201, 33))
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(14)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)
        self.label.setFont(font)
        self.label.setStyleSheet("font: 14pt \"MS Shell Dlg 2\";")
        self.label.setObjectName("label")
        self.finalscore = QtWidgets.QLabel(Dialog)
        self.finalscore.setGeometry(QtCore.QRect(110, 120, 131, 41))
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(18)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)
        self.finalscore.setFont(font)
        self.finalscore.setStyleSheet("color: rgb(129, 129, 193);\n"
"\n"
"font: 18pt \"MS Shell Dlg 2\";")
        self.finalscore.setAlignment(QtCore.Qt.AlignCenter)
        self.finalscore.setObjectName("finalscore")
```

```
        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.label.setText(_translate("Dialog", "Your Team Score :"))
        self.finalscore.setText(_translate("Dialog", "0"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

RESULT OF PYTHON PROJECT

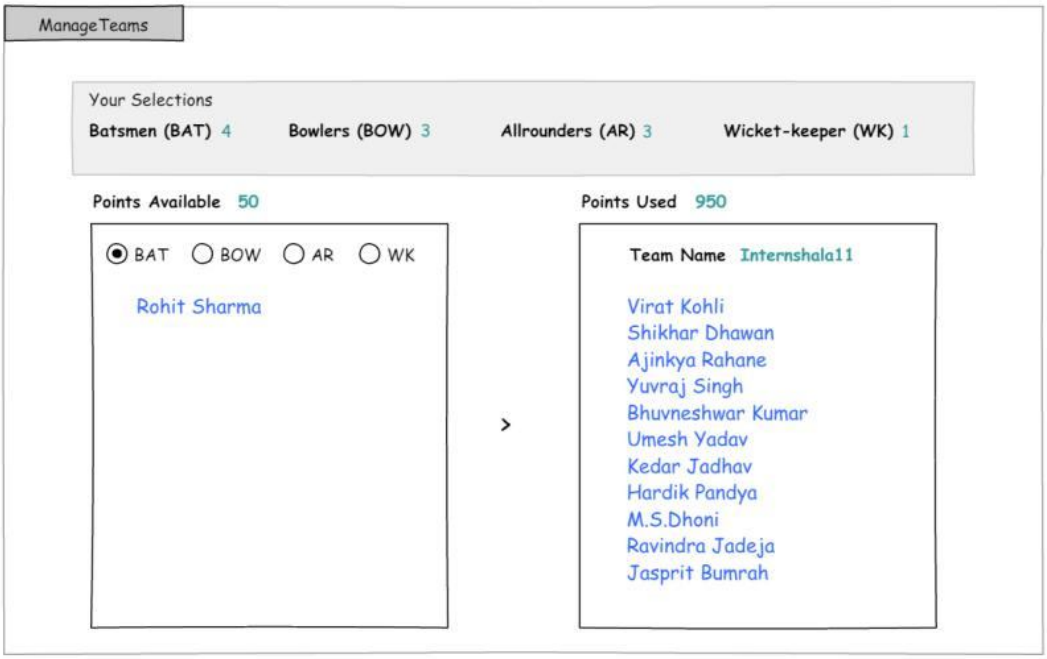


1 ***- Opening screen of the application. You can see the players of each category by selecting the category. To begin with, the selection is disabled until a new team is created from the Manage Teams menu. A pop up asking the name of the team appears.***

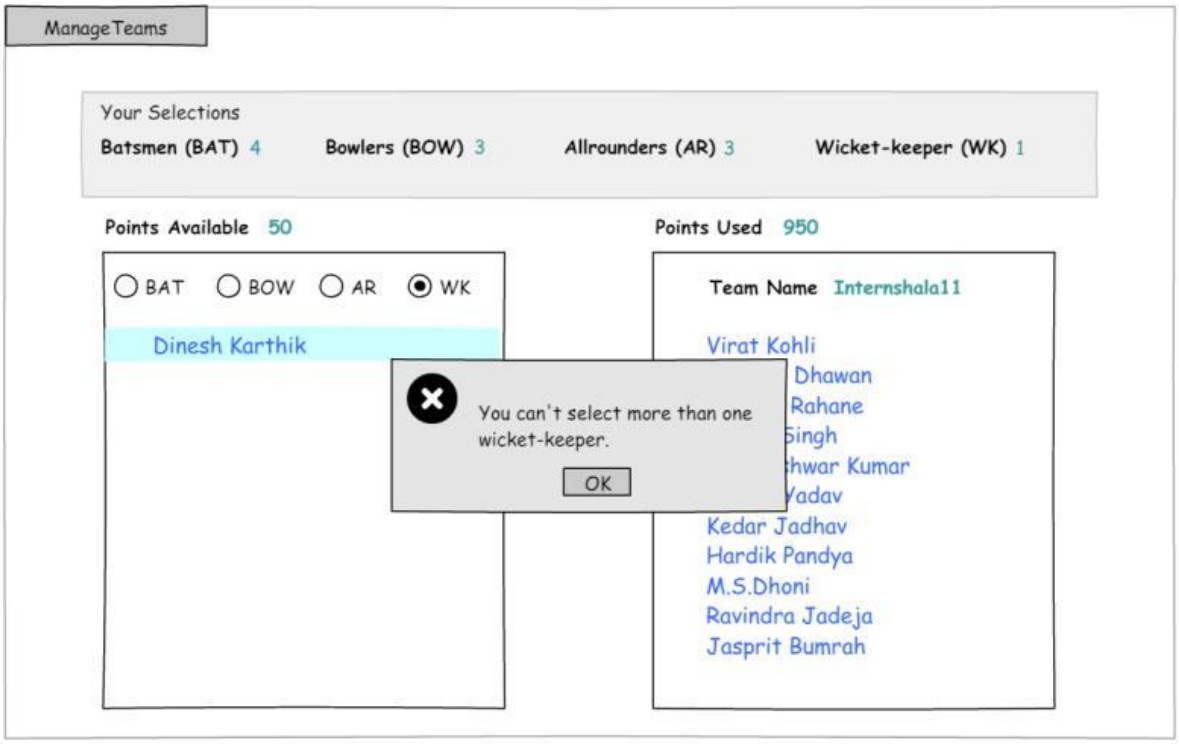


2 ***- The toolbar menu options which allow you to create a new team, open an existing team, save your team and finally evaluate the score of a saved team.***

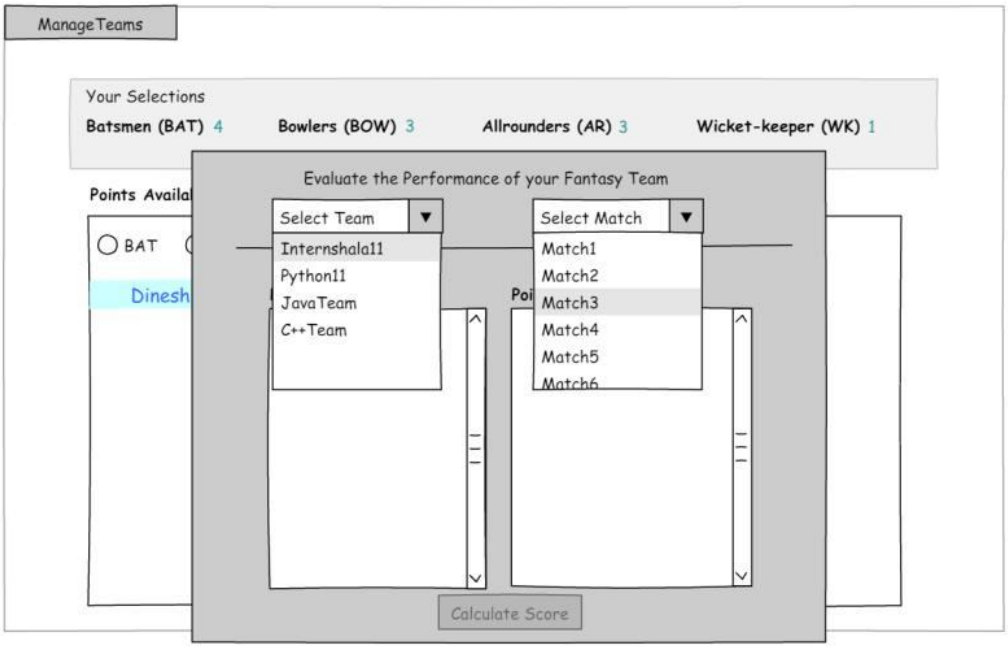
3 ***- After clicking New Team, the left box is populated with player names. As you select a different category, the corresponding list of players is displayed.***



4 ***- On double-clicking each player name, the right box gets populated. Points available and used are displayed accordingly.***



*5 - **Message if the game logic is not followed***



*6 - **Pop-up on clicking Evaluate Score. You can select your team here and the match for which the players' performance is compared.***

7 - The final score for your fantasy team based on the match selected.

FUTURE SCOPE

Application of Python in Various Disciplines

Hello folks!

Congratulations on completing the six learning modules of this training. All the essential aspects of Python programming were covered in the six modules. I hope you enjoyed this training as much as I did bringing it to you.

Application of Python in Various Disciplines

Data Science

In recent times, Python has shot up in popularity charts mainly because of its Data science libraries. With huge amount of data being generated by web applications, mobile applications and other devices, companies need business insights from this data.

Today Python has become language of choice for data scientists. Python libraries like NumPy, Pandas and Matplotlib are used extensively in the process of data analysis and collection, processing and cleansing of data sets, applying mathematical algorithms to data and to generate visualizations for the benefit of users. Commercial Python distributions such as Anaconda and Activestate provide all the essential libraries required for data science.

Data science today has become a prominent buzzword and it has been termed as ‘sexiest job of 21st century!’.

Machine Learning

This is another glamorous application area where Python developers are getting attracted. Based upon the past data, Python libraries such as Scikit-learn, Tensorflow, and NLTK are widely used for prediction of trends like customer satisfaction, projected values of stocks etc.

Some of the real world applications of machine learning are as under:

Medical Diagnosis: Machine learning techniques are used for the analysis of the importance of clinical parameters and of their combinations for prognosis, e.g. prediction of disease progression, for the extraction of medical knowledge for outcomes research, for therapy planning and support, and for overall patient management.

Statistical Arbitrage: Machine learning methods are applied to automate trading strategies where user tries to implement a trading algorithm for a set of securities on the basis of quantities such as historical correlations and general economic variables.

Learning associations: Machine learning helps the process of developing insights into various associations between products and buying behaviors of customers.

Basket analysis- studying the association between the products people buy and suggesting the associated product to the customer, is a well known phenomenon we see while doing online shopping. Machine learning is at work behind this analysis.

Prediction: Current prediction is one of the hottest machine learning algorithms. Businesses are interested in finding out what will be my sales next month / year / Diwali, etc. so that business can take required decision (related to procurement, stocks, etc.) on time.

Web Development

Another application area which is becoming increasing popular with Python developers is web development. Simple to complex web applications can be developed using easy to use web application frameworks like django, Pyramid, Flask etc. These frameworks are used extensively by various IT companies. Dropbox for example uses django as a backend to store, synchronize local folders.

Most of the web servers today are compatible with WSGI (Web Server Gateway Interface) – a specification for universal interface between Python web frameworks and web servers. All leading web servers such as Apache, IIS, Nginx etc can now host Python web applications. Google's App Engine hosts web applications built with almost all Python web frameworks.

Image processing

Face detection and gesture recognition using OpenCV library and Python is another important application. OpenCV is a C++ library, but has been ported to Python. This library has lot of extremely powerful image processing functions.

Facebook's automatic tag suggestion feature, which used face recognition to suggest people you might want to tag in your photos, is an instance of OpenCV at work. Other instances of OpenCV applications are:

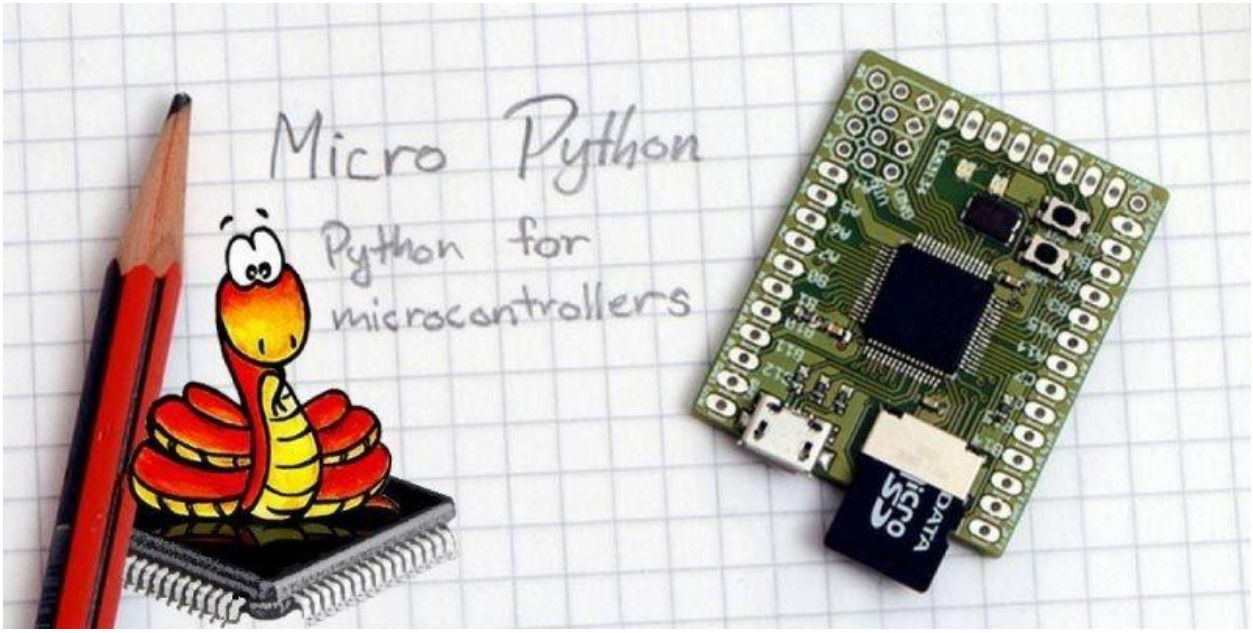
- Automated inspection and surveillance.
- Robot and driverless car navigation and control.
- Medical image analysis.
- Video/image search and retrieval.

Game Development

Python is a popular choice of game developers. The Pygame library is extensively used for building games for desktop as well as mobile platforms. Pygame applications can be installed on Android too.

Embedded Systems and IoT

Another important area of Python application is in embedded systems. Raspberry Pi is a very popular yet a low cost single board computer. It is being extensively used in automation products, robotics, IoT, and kiosk applications. Those of you with Electronics background can certainly explore the possibility of becoming innovators.



Apart from Raspberry Pi, other microcontrollers can also be programmed with Python. A lightweight version of Python called micropython has been developed especially for microcontrollers. Popular microcontrollers like Arduino are used in many IoT products and programmed with Python. A special micropython compatible controller called Pyboard has also been developed.

Android apps

Although Android apps are predominantly developed using Android SDK which is more or less like Java, Python can also be used to develop Android apps. Python's Kivy library has all the functionality required to build a mobile application.

Automated Jobs

Python is extremely useful and widely used for automating CRON jobs. Certain tasks like backups can be scheduled to be invoked automatically. These tasks are defined in Python scripts and operating system scheduler executes them at predefined times. Python is embedded as a scripting language in many popular software products. This is similar to VBA used for writing macros in Excel, Powerpoint etc. Similarly Python API is integrated with Maya, PaintShop pro, etc.

Python as a Rapid Development Tool

Although Python is becoming increasingly popular, software development field is still dominated by Java and Dot net platform. However, Java and C# developers are also taking help of power of Python's rapid development capabilities. Jython, the JRE implementation of Python and IronPython, the dot net implementation interact seamlessly with Java and C# respectively. For example- Jython can use all Java libraries such as Swing, etc. So the development time can be minimized by using simpler Python syntax and Java libraries for prototyping of the software product.

Python has found applications in very diverse fields. There is Astropy library is a collection of software packages written in the Python programming language and designed for use in astronomy. BioPython package contains functionality useful for computational biology and bio-informatics.

More and more companies, be it start ups or established ones are using Python for one or other purpose in their development activities. Hence, lot of opportunities are waiting for those with appropriate Python skills. This training program has given a solid foundation for you to grab these opportunities. Internshala wishes you all the best in your endeavour!

REFERENCE

- trainings.internshala.com
- www.instamojo.com
- www.google.com
- www.vinfotech.com
- www.scribd.com
- **http://python.org**
- **http://diveintopython.org**
- **http://djangoproject.com/**