# 1. Introduction to Python

Python is a popular, high-performance programming language. It was created in 1991 by Guido van Rossum and is maintained by the Python Software Foundation. Its syntax allows programmers to convey concepts in a few lines of code, and it is intended primarily to stress code readiness. Python is a programming language that helps you to work quicker and more efficiently integrate systems. Python is a text language that is high-quality, translated, interactive, and focused. Python is intended to be extremely readable. It employs English terms more frequently than other languages, yet it has a lesser grammatical structure than other languages. To make maintenance and upgrades simpler, you should focus on the quality of the source code while creating a software application. Syntax rules in Python allow you to express concepts without having to write any additional code.

In contrast to other programming languages, Python prioritizes readability and allows you to utilize English terms instead of punctuation. As a result, you may construct bespoke apps using Python without having to write any more code. A legible and clean code base will save you time and effort in maintaining and updating the product. Python, like other current programming languages, allows for multiple editing. Supports a completely organized and object-focused system. In a functional and side-oriented system, its language characteristics also support many notions. Python also has a memory management framework that is both flexible and methodical. Python's editing paradigms and language capabilities aid in the improvement of big and complicated applications.

Python now supports a variety of applications. Python translators can even be used to adapt the code to certain platforms and tools. Python is also a language that can be translated. Allows you to reuse code across different platforms without having to reconstruct it. As a result, you won't have to reassemble the code after making any changes. You may use the application's updated code without reversing it, and you can see the results of the changes right away. The functionality allows you to make changes to the code without slowing down the upgrading process.

Python outperforms all other programming languages due to its huge and robust standard library. Depending on your needs, you may pick from a range of modules in the standard library. Each module allows you to extend the functionality of a Python program without having to write new code. You may use various modules to execute web services, conduct cable activities, handle system interface, and work via online contracts, for example, when creating a web application in Python. You may also learn about other modules by looking through the Python Standard Library docs.

Python, being an open-source programming language, may help you save a lot of money on software development. You may also leverage a few free source Python frameworks, libraries, and development tools to speed up development without raising expenditures. Depending on your individual demands, you may also select from a variety of open Python source frameworks and development tools. Python's strong web frameworks, such as Django, Flask, Pyramid, Bottle, and Cherrypy, for example, can ease and expedite online application development. Similarly, Python GUI frameworks and toolkits such as PyQT, PyJs, PyGUI, Kivy, PyGTK, and WxPython can help you construct desktop GUI applications faster.

Python is a well-known and widely used programming language. As a result, programming languages may be used to improve both desktop and online apps. Python may also be used to create complicated scientific and math applications. Python has capabilities that aid data analysis and visibility. You can use Python's data analysis tools to generate fantastic bespoke data solutions without putting in a lot of

work. At the same time, Python's visualization packages and APIs assist you in visualizing and presenting data in an appealing and effective manner. Python is also widely used to create artificial intelligence (AI) and natural language processing features by many Python programmers.

Python may be used to construct a software prototype in seconds. Also, by rewriting the Python code, you may create a software application directly on the prototype. Using the test-based development (TDD) technique, Python makes it simple to code and test at the same time. You can quickly write the necessary tests before developing the code, and you can utilize the tests to regularly examine the application code. The test may also be used to see if an application's source code fulfils the pre-defined requirements.

Python, like other programming languages, is not without problems. It lacks some of the built-in functionality of more current editing languages. To speed up bespoke software development, you should leverage Python libraries, modules, and frameworks. Python is also slower than the most extensively used programming languages, such as Java and C++, according to a few studies. You must speed up the Python program by altering the code or employing custom operating time. Python, on the other hand, may be used to speed up program development and make software maintenance easier.

**Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

**Python is Interactive** − You can 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.

## 1.2 Brief report of the goal

Python allows you to create your ideal cricket match. All the features in the game must be presented on the false displays in the mode. We may use rules like the sample rules to compute each player's score.

**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

## 2. History

Guido van Rossum invented Python at the Netherlands' National Research Institute for Mathematics and Computer Science in the late 1980s and early 1990s. Python is built on several different languages, including ABC, Modula-3, C, C ++, Algol-68, SmallTalk, Unix shell, and other scripting languages. Python is a trademarked language. Python source code, like Perl, is currently accessible under the GNU General Public License (GPL). Python is now maintained by the center's primary development team; however, Guido van Rossum has been instrumental in driving its evolution.

History was going to be written in the late 1980s. Python got to work at that point. Guido Van Rossum began working at Centrum Wiskunde & Informatica (CWI) in the Netherlands shortly after, in December 1989. It began as a pastime because he desired an intriguing endeavor to keep him occupied throughout the Christmas season. Python programming was a popular ABC programming language with ties to the Amoeba Operating System and its own set of capabilities. At the start of his career, he was already working to construct ABC and had experienced issues with it, but he enjoyed many elements of it. Then he pretended to be intelligent. He'd taken the ABC syntax, as well as some of its finer points. There were

also a lot of complaints, so he handled them all and produced a nice writing style that was error-free. The name was inspired by the BBC TV program 'Flying Circus by Monty Python,' which she was a great admirer of and wanted a short, creative, and ambiguous name for, so she devised Python! He was a "benevolent dictator for life" (BDFL) until July 12, 2018, when he stood down as leader. He formerly worked for Google, but now he is employed by Dropbox.

In 1991, the language was eventually launched. In comparison to Java, C++, and C, it utilized extremely few codes to convey ideas when it was first published. Its design philosophy was also excellent. Its major goal is to make code more readable and to help programmers succeed in their careers. It had more than enough capability when it was first published to offer classes inheritance, a few sorts of master data without management, and operations.

## 3. Python Features

**Names, Scopes and Binding**

In Python, namespace is a naming scheme that gives everything a unique name. The object might be a method or a modification. Python keeps track of words in the form of a dictionary. Let's look at the architecture of a text file system on a computer, for example. Many texts with the same file name can be found throughout the index. However, if you offer a whole file path, you can be led to that file.

Similarly, depending on the word spacing, the Python translator recognizes whether the user is attempting to point out a clear or flexible method in the code. As a result, the word's separation gives additional information. The name can be any Python or flexibility method, and the space is determined by where the variables or method are to be accessed.

- The width of the variable is the width of the statements in which the variable is displayed.
- The exception is apparent in the statement if it can be referred to in that statement.
- The local variable is local to the system unit or block when announced there.
- Non-local variables of a program unit or block are those that appear within the system unit or block but may not be announced there.

**Data Types:**

The categorization or classification of data items is known as data types. Represents a value that specifies which tasks may be done on a given set of data. The data types are classes, and the variables are an example (object) of these classes, because everything in the Python system is an object.

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary

**Numeric:**

Numeric data is represented by a kind of numerical data in Python. A whole integer, a floating number, or a complex number can be used as the numeric value. In Python, these values are represented by the int, float, and complex classes.

The whole amount is represented by the int class. It's chock-full of positive and negative numbers (excluding friction or decimal). There is no limit on how lengthy a number may be in Python.

Float - The float category represents this value. A floating point is represented by a real number. The decimal point was defined. To explain the scientific statement, add the letter e or E followed by a positive or negative number if appropriate.

Complex categories - A complex category represents a complex number. (Actual portion) + (imaginary part) j is the specification.

**Sequence Type:**

A sequence in Python is an ordered collection of the same or various data types. You may use sequence to hold several values in an organized and efficient manner. In Python, there are numerous different forms of sequences:

- String
- List
- Tuple

**1) String**

Cables are a list of bytes in Python that represent Unicode characters. In a single quotation, double quote, or triple quote, a character unit is a group of one or more letters. There is no character data type in Python; instead, a character is a sequence of one length. Represented by the section of str.

**2) List**

The list is like the draft, which is an organized data set that has been announced in various languages. Because the objects mentioned do not have to be of the same kind, it is very flexible.

**3) Tuple**

Tuple is an ordered collection of Python objects in the form of a list. The sole difference between a tuple and a list is that tuples are immutable, meaning they can't be modified after they've been created.

**Boolean:**

True or False are the two built-in values for this type of data. Things that are reasonable and equal to the Truth are true (true), while those that are reasonable and equal to the False are false (false). Non-existent objects, on the other hand, can be checked and judged to be true or false in a Boolean context. The class bool is used to display this information.

**Set:**

Set is a Python term that refers to a random collection of useable, dynamic data that has no duplicate features. Although it may incorporate numerous factors, the sequence of the items in the set is not stated.

**Dictionary:**

Python's dictionary is a random collection of data values that may be used to store data values as a map. Unlike other Data Types, Dictionary has key: value pairing. To make it even better, the dictionary includes the key value. A colon: separates each pair of terms in the dictionary, whereas a 'comma' separates each key.

**Control Flow Statements:**

The performance of your loop control statements differs from that of your usual sequence. All of the default features developed in that range are removed when the executions leave a trace. The following control statements are supported by Python.

**Sub-program:**

- There is just one entry point for each sub-program.
- The caller is suspended while a small system called is running, which implies that only one small system is running at any given moment.
- When the usage of a little program called termination is terminated, control is always restored to the caller.

**Paradigms Supported:**

Priority, functionality, process, and focus are the four basic editing paradigms supported by Python. Python aims to make the zones available and usable, whether you believe that they are practical or beneficial. Before we go into whether editing paradigm is best for certain application scenarios, let's take a short look at what we've learned so far.

**Easy to read** - Python has a few keywords, a simple structure, and a clearly defined syntax. This allows the reader to quickly grasp the language.

**Easy to read** - Python code is clearly defined and visual.

**Easy storage** - Python source code is easy to store.

**General Library** - Python bulk library is very portable and compatible with various platforms at UNIX, Windows, and Macintosh.

**Interactive Mode** - Python has interactive mode support that allows interactive exploration and correction of code caption errors.

**Portable** - Python can work on a variety of hardware platforms and has the same connectivity across all platforms.

**Extensible** - You can add sub-modules to a Python translator. These modules allow programmers to add or customize their tools to work more efficiently.

**Database** - Python provides interactive platforms for all major commercial websites.

**GUI Programming** - Python supports GUI applications that can be created and delivered to multiple system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and X Window system Unix.

**Scalable** - Python provides better structure and support for larger programs than shell text. In addition to the features listed above, Python has a huge list of great features, just a few of which are listed below.

It Supports efficient and systematic planning methods with OOP.

It can be used as a writing language or can be integrated into a bytecode to create great applications.

It provides high-quality data types and supports dynamic type testing.

IT supports automatic garbage collection.

It can be easily integrated with C, C ++, COM, ActiveX, CORBA, and Java.

## 4. Distinctive Language Features:

There are several editing languages available today, each with its own features. Choosing the proper planning language for the project is the most significant element that aids planning. As a result, before selecting which is ideal for you, you should learn Python programming. Knowing the language also means knowing the features. Below are some of the features that will help you understand why Python is better than R or other roots.

- Easy to code and Read
- High-Level Programming Language
- Portable
- Expressive
- Object Oriented
- Free and Open Source

- Interpreted
- Extensible
- Embeddable
- Large and Standard Library
- GUI programming
- Dynamically Typed

## 5. Tutorial:

Python is a strong and easy-to-understand programming language. It offers high-quality data structures as well as a straightforward yet effective object-focused editing technique. Python's great syntax and dynamic typing, as well as its translation environment, make it a good language for rapid scripting and application development on a variety of platforms.

Python Translator and the Python general library are available for free in source or binary form from the Python website, https://www.python.org/, and can be freely distributed. There are also distributions and references to numerous free Python modules, applications, and tools, as well as extra documentation, on the same site.

Python Translator may simply be extended with additional C or C++ functions and data types (or other C-driven languages). Python may also be used as an extension language for developing bespoke apps.

This course provides an informal introduction to the fundamental principles and features of the Python language and environment. For further detail, a decent Python interpreter is helpful, although all examples are self-contained, so the lesson may also be read offline.

See the Python Standard Library for a list of common features and modules. The official definition of Python is found in the Python Language Reference. Read the Extensions and Embedding Python Interpreter and the Python / C API Reference Manual to learn how to develop extensions in C or C++. There are also a few books that go into detail about Python.

This course does not aim to cover every feature or even all the more popular ones. Instead, it teaches many of Python's most noteworthy features, giving you a clear sense of the language's taste and style. After reading it, you'll be able to read and write Python modules and programs, and you'll be ready to learn more about The Python Standard Library's many Python library modules.

**1. Python Introduction**

Learn how to install Python, distinguish between important types of data, and use the basic features of the Python interpreter, IDLE.

**2. Using Variable in Python**

Learn about numbers, alphabet unit, sequence and types of dictionary data and relevant functions while running Python syntax.

**3. Editing Basics on Python**

Learn to write programs using scenarios, loops, iterators and generators, functions and modules and packages.

**4. Principles of Focus Program (OOP)**

Learn about the key features of Object-oriented Planning while using Classes and Objects, two main features of the OOP paradigm.

**5. Links to SQLite Database**

Learn about the relationship site while learning to store and retrieve data from the SQLite website via Python.

**6. Upgrade GUI with PyQT**

Learn how to install PyQt5 toolkit, Qt Designer and create user interface using custom widgets and menu programs.

**7. The Use of Python in Various Studies**

Learn about various resources to expand your learning with Python editing language.

**Sample Program:**

A simple program featuring "Hello, Earth!". It is often used to indicate language syntax.

```
# This program prints Hello, world!

print('Hello, world!')
```

# 6. Evaluation

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

**Match 1**

| Player | Score | Balls Faced | Fours | Sixes | Bowled | Maiden | Given | Wickets | Catches | Stumping | RO* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |

***Run Out**

**Team**

| Name | Players |
|---|---|
| | |

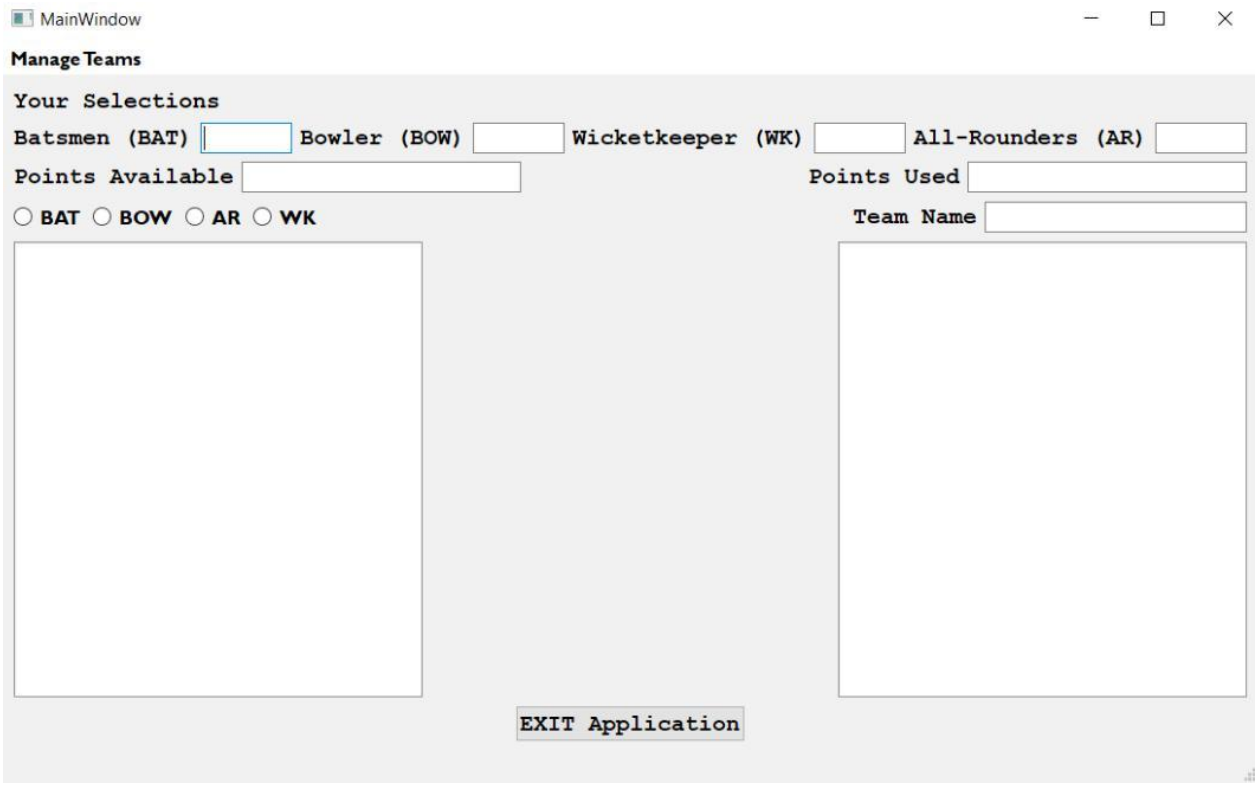
**Stats**

| Player | Matches | Runs | 100s | 50s | Value | Category |
|---|---|---|---|---|---|---|
| | | | | | | |

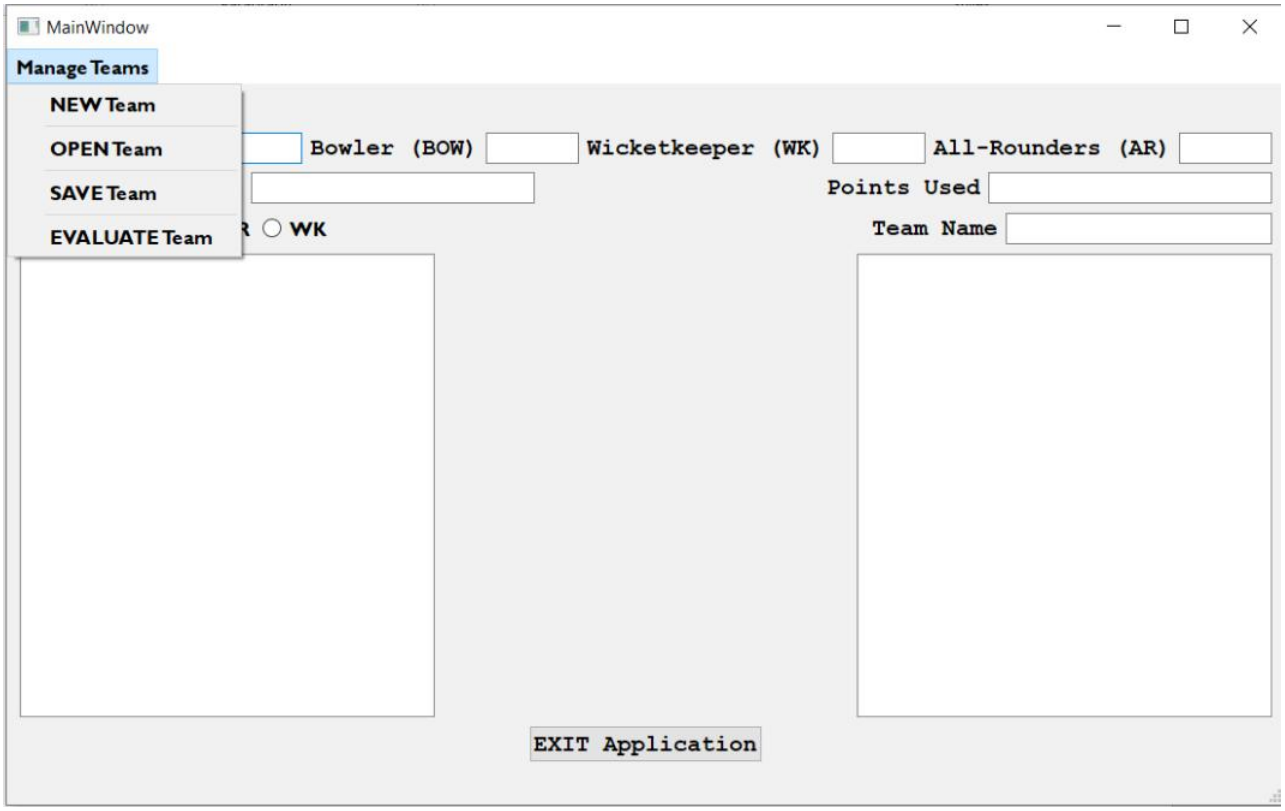
**The data to enter in the remaining two tables is given below:**

| 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 |

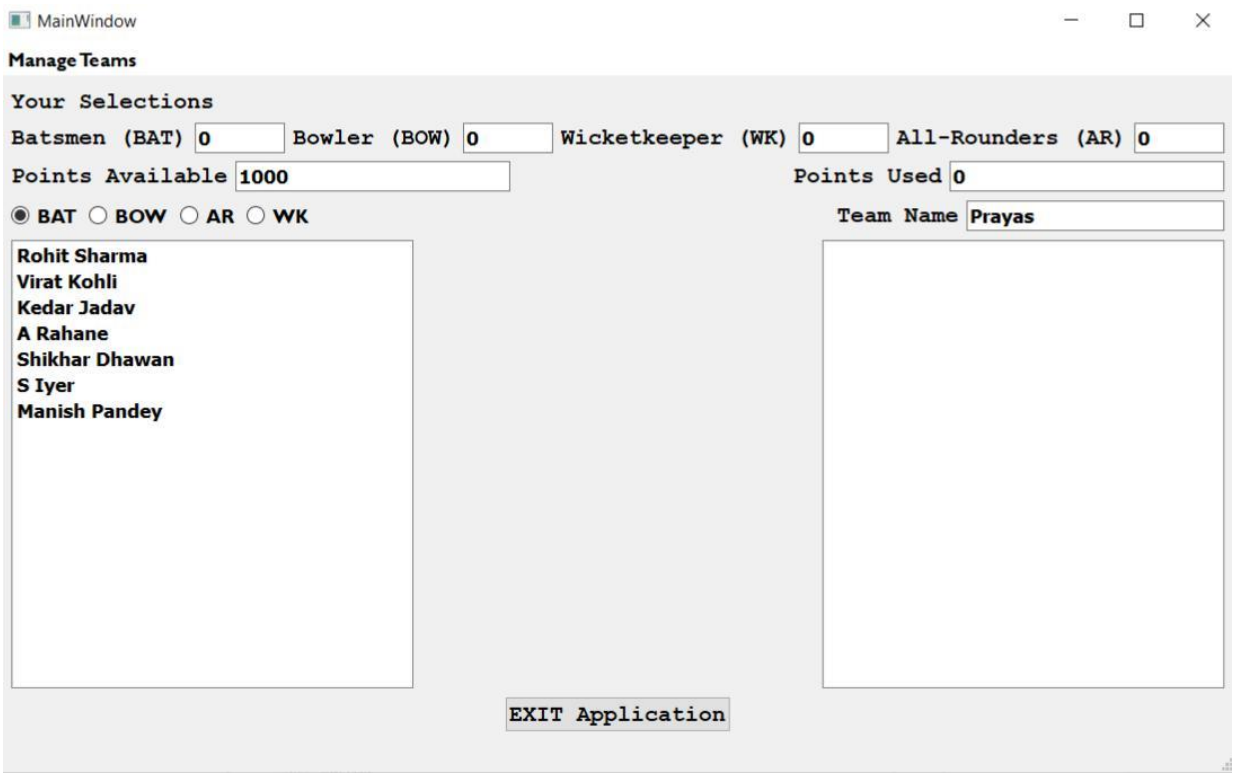
## Testing



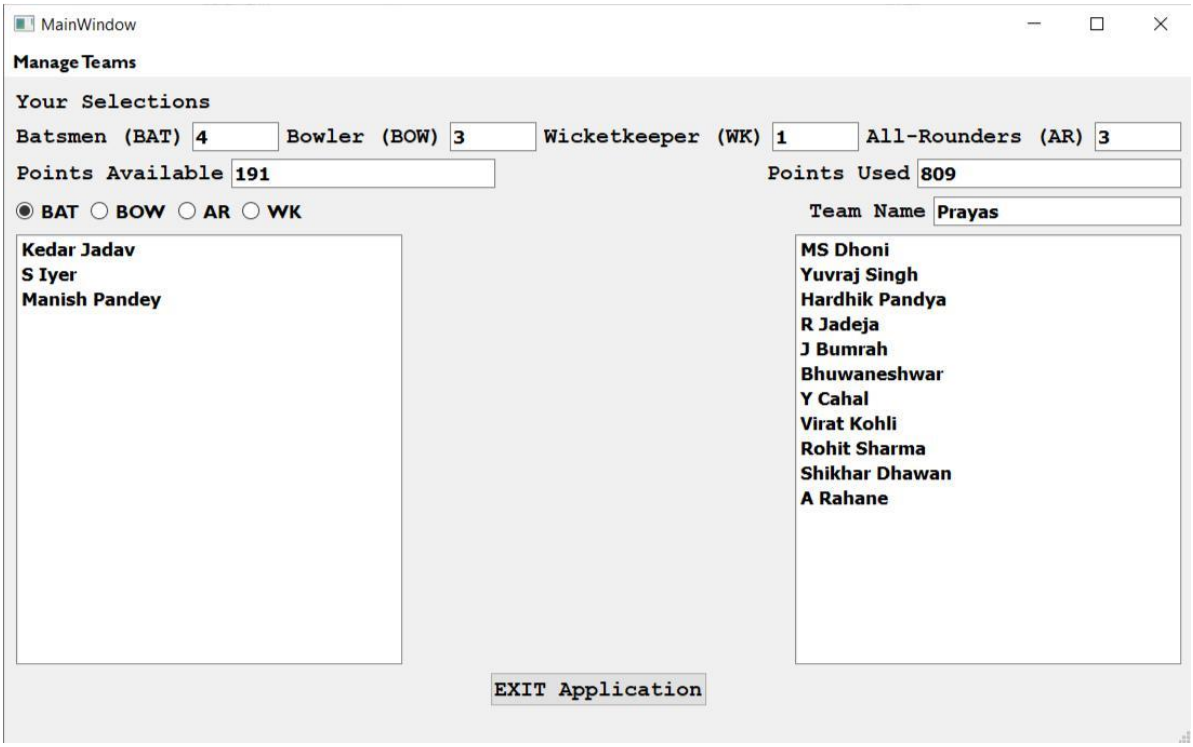

Opening screen of the application. You can see the players of each category by selecting the category. To begin with, the selection is disabled until a new team is created from the Manage Teams menu. A pop up asking the name of the team appears.



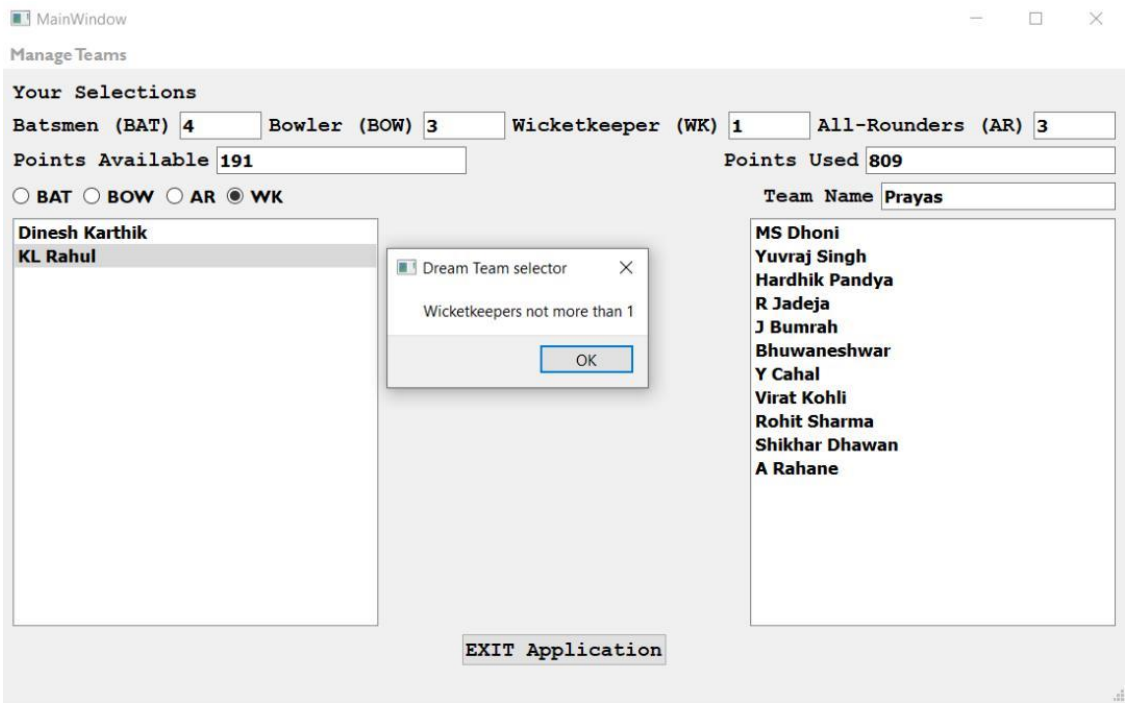

The toolbar menu options which allow you to create a new team, open an existing team, save your team, and finally evaluate the score of a saved team.



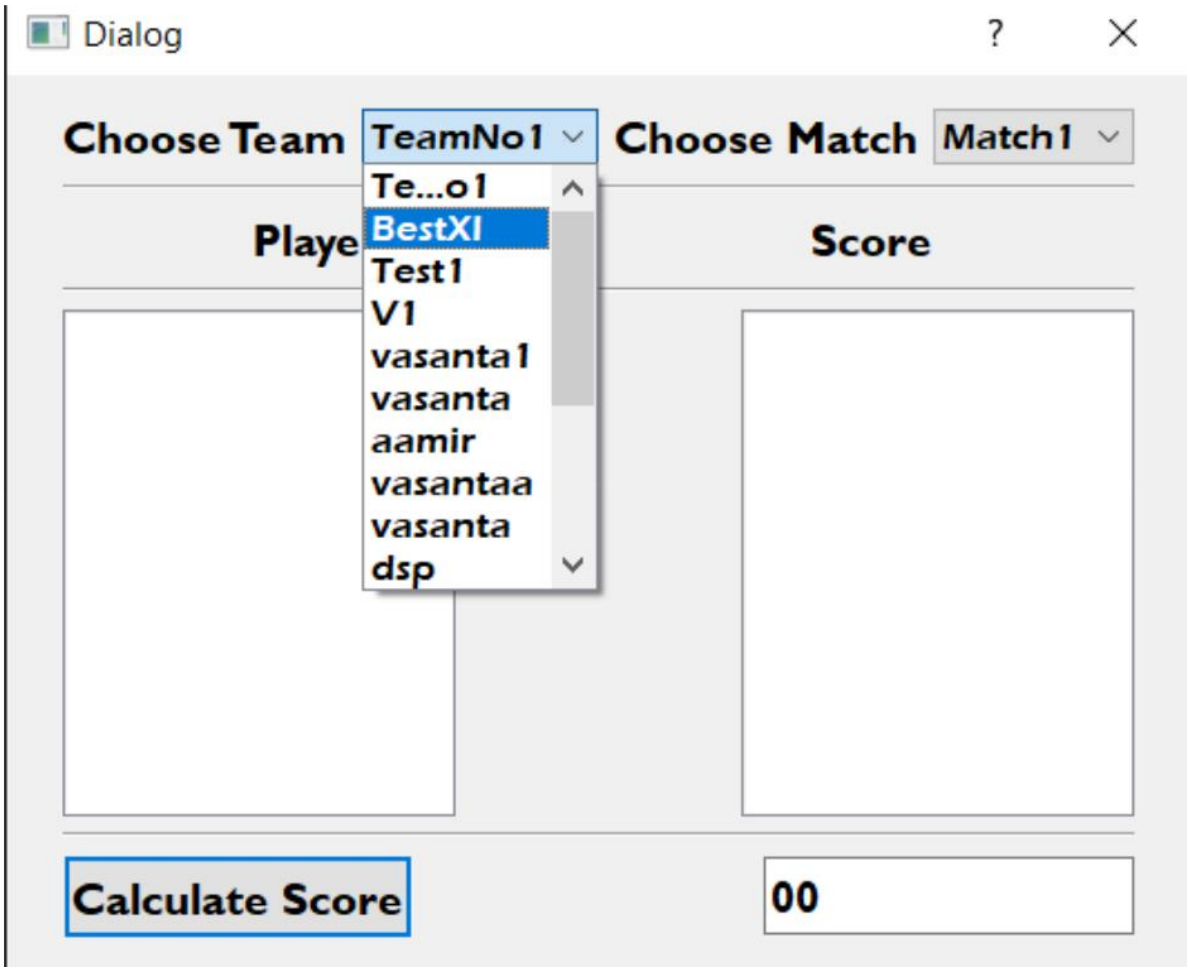
After clicking New Team, the left box is populated with player names. As you select a different category, the corresponding list of players is displayed.



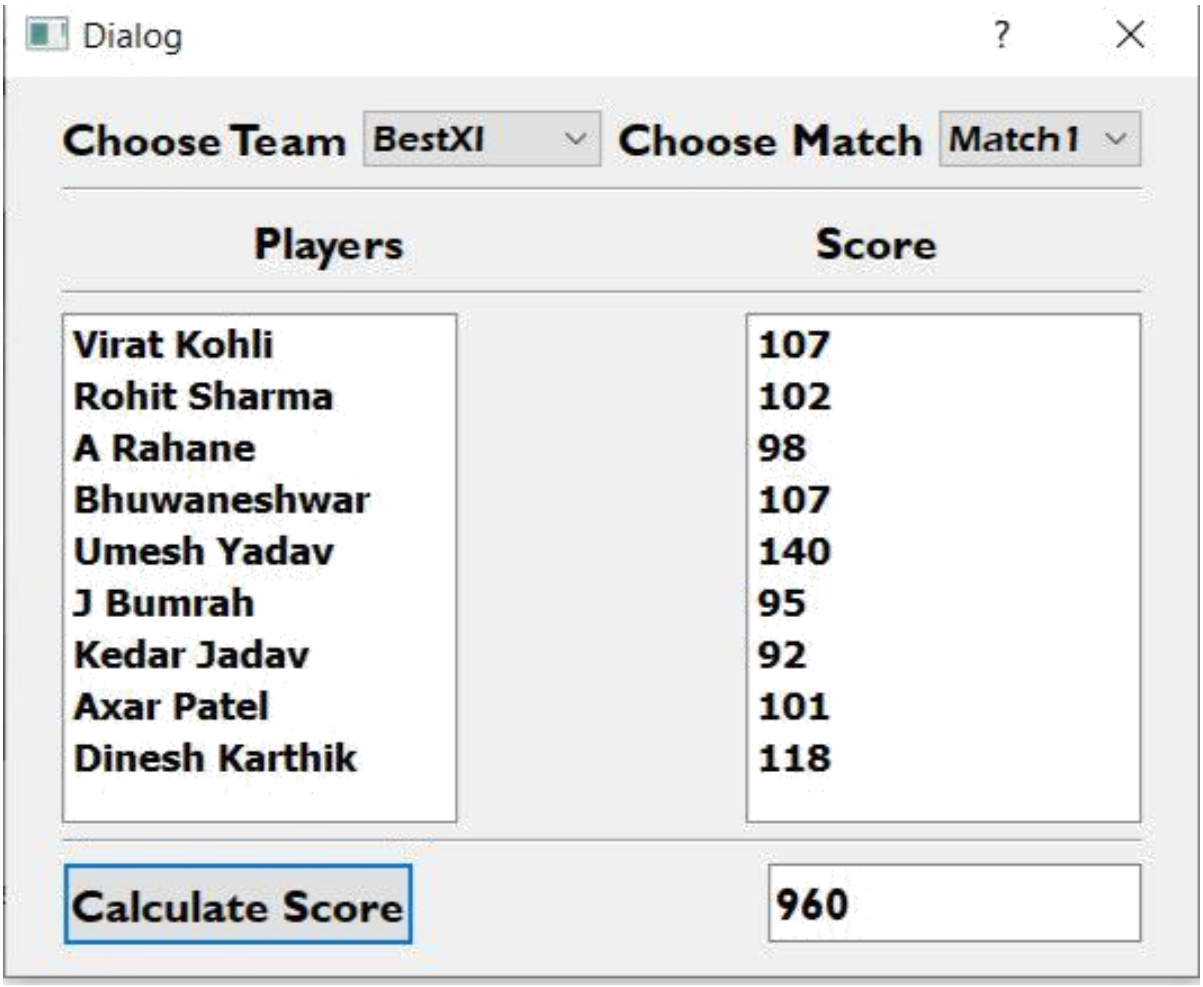

On double-clicking each player name, the right box gets populated. Points available and used are displayed accordingly.



Message if the game logic is not followed

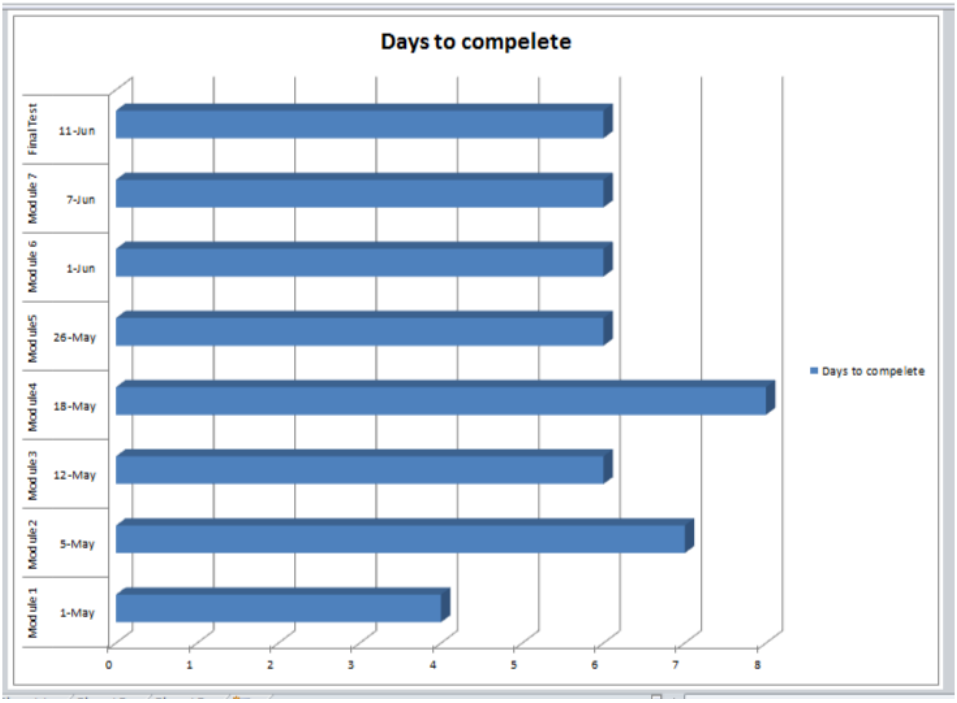



Upon opening the second file to evaluate the scores. You can select your team here and the match for which the players' performance is compared.




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

## Gantt chart




**Problem Analysis**

**PRODUCT DEFINITION: -**It is a game where you create a 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.

**FEASIBILITY ANALYSIS: -** I am building a software for gaming purposes using a specific technology named python. It is a game software where you can create virtual team according to your choice and score points to win a tournament.

This software is created for motivating street cricket and adding more fun and entertainment to cricket. The components that are used in this demo can be integrated to a high extent to provide statics to different components of cricket. This project helps in providing real time on field actions there by helping its user of the current actions happening on field.

**Coding**

File Edit Format Run Options Window Help

```
    def menu(self,action):
        txt=(action.text())

        if txt=='NEW Team':
            self.bat=0
            self.bwl=0
            self.ar=0
            self.wk=0
            self.avl=1000
            self.used=0
            self.list1.clear()
            self.list2.clear()
            #self.t7.setText("???")

            #self.dial()
            text, ok=QtWidgets.QInputDialog.getText(MainWindow, "Team", "Enter name of team
            if ok:
                self.t7.setText(str(text))

            self.show()


        if txt=='SAVE Team':
            count=self.list2.count()
            selected=""
            for i in range(count):
                selected+=self.list2.item(i).text()
                if i<count:
                    selected+=","
            self.saveteam(self.t7.text(),selected,self.used)

        if txt=='OPEN Team':
            self.bat=0
            self.bwl=0
            self.ar=0
            self.wk=0
            self.avl=1000
            self.used=0
            self.list1.clear()
            self.list2.clear()
            self.show()
            #print("rgr")
            self.openteam()


        if txt=='EVALUATE Team':
            from dlgscore import Ui_Dialog
```

Ln: 334 Col: 27

File Edit Format Run Options Window Help

```
    def saveteam(self,nm,ply,val):
        #print("rvrv")

        if self.bat+self.bwl+self.ar+self.wk!=11:
            self.showdlg("Insufficient players")
            return

        #print("frbfj")
        sql="INSERT INTO teams (name,players,value) VALUES ('"+nm+"','"+ply+"','"+str(val)+'
        #print("f3f4")
        try:
            #print("bjr")
            cur=conn.execute(sql)
            #print("dehe")
            self.showdlg("Team Saved Succesfully")
            conn.commit()
        except:
            self.showdlg("Error in Operation")
            conn.rollback()


    def showdlg(self,msg):
        #print("ecb")
        Dialog=QtWidgets.QMessageBox()
        Dialog.setText(msg)
        Dialog.setWindowTitle("Dream Team selector")
        ret=Dialog.exec()


if __name__ == "__main__":
    import sqlite3
    conn = sqlite3.connect('fantasy.db')
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Ln: 546 Col: 4

File Edit Format Run Options Window Help

```
    def show(self):
        #print("vvrv")
        self.t1.setText(str(self.bat))
        self.t2.setText(str(self.bwl))
        self.t3.setText(str(self.wk))
        self.t4.setText(str(self.ar))
        #print("rrrr")
        self.t5.setText(str(self.avl))
        self.t6.setText(str(self.used))
        #print("efef")



    def criteria(self,ctgr,item):
        msg=''
        if ctgr=="BAT" and self.bat>=5:msg="Batsmen not more than 5"
        if ctgr=="BWL" and self.bwl>=5:msg="bowlers not more than 5"
        if ctgr=="AR" and self.ar>=3:msg="Allrounders not more than 3"
        if ctgr=="WK" and self.wk>=1:msg="Wicketkeepers not more than 1"
        if msg!='':
            #msg="You Have Exhausted your Points"
            self.showdlg(msg)
            return False

        if self.avl<=0:
            msg="You Have Exhausted your Points"
            self.showdlg(msg)
            return False

        if ctgr=="BAT":self.bat+=1
        if ctgr=="BWL":self.bwl+=1
        if ctgr=="AR":self.ar+=1
        if ctgr=="WK":self.wk+=1


        sql="SELECT value from stats where player='"+item.text()+"'"
        cur=conn.execute(sql)
        row=cur.fetchone()
        self.avl-=int(row[0])
        self.used+=int(row[0])
        return True
```

Ln: 546 Col: 4

File Edit Format Run Options Window Help

```
    def calculate(self):
        import sqlite3
        conn = sqlite3.connect('fantasy.db')
        team=self.cb0.currentText()
        self.lw1.clear()
        sql1="select players, value from teams where name='"+team+"'"
        cur=conn.execute(sql1)
        row=cur.fetchone()
        #print (row[0])
        selected=row[0].split(',')
        #print (selected)
        self.lw1.addItems(selected)
        teamttl=0
        #print("ded")
        self.lw2.clear()
        match=self.cb1.currentText()
        for i in range(self.lw1.count()):
            ttl, batscore, bowlscore, fieldscore=0,0,0,0
            nm=self.lw1.item(i).text()
            cursor=conn.execute("select * from "+match+" where player='"+nm+"'")
            row=cursor.fetchone()
            #print ("fekn")
            #print(row)
            batscore=80
            if batscore>=50: batscore+=5
            if batscore>=100: batscore+=10
            if row[1]>0:
                sr=row[1]/row[2]
                if sr>=80 and sr<100: batscore+=2
                if sr>=100:batscore+=4
            batscore=batscore+row[3]
            batscore=batscore+2*row[4]
            #print ("batting score=", batscore)
            bowlscore=row[8]*10
            if row[8]>=3: bowlscore=bowlscore+5
            if row[8]>=5: bowlscore=bowlscore=bowlscore+10
            if row[7]>0:
                er=6*row[7]/row[5]
                #print ("eco:")
                if er<=2: bowlscore=bowlscore+10
                if er>2 and er<=3.5: bowlscore=bowlscore+7
                if er>3.5 and er<=4.5: bowlscore=bowlscore+4
            fieldscore=(row[9]+row[10]+row[11])*10
```

Ln: 226 Col: 12

**Reference:**


https://www.w3schools.com/python/

https://wiki.python.org/moin/PyQt/Tutorials

https://www.tutorialspoint.com/pyqt/

https://www.tutorialspoint.com/sqlite/sqlite_quick_guide.htm

https://www.quora.com/

https://www.programiz.com/python-programming/examples/hello-world

https://docs.python.org/3/tutorial/