IQRA UNIVERSITY IU

Laboratory Workbook

Visual Software Analytics

Section- TUE (11:45/1:45)
Program: BS-SE
Instructor Name: Sir Tayyab Gul

Group Members

S. No	**Roll Number**	**Name**
1	37888	Haris Bin Irfan
2	37728	Yasir Ul Haq
3	38633	Syed Usama Ahmed Hashmi

Introduction

In this MOOC, we explore how the effectiveness of software development projects can be pro-actively improved by applying concepts, techniques, and tools from software diagnosis.

The term "software diagnosis" refers to recently innovated techniques from automated software analysis and software visual analytics that aim at giving insights into information about complex software system implementations, the correlated software development processes, and the system evolution.

To this end, all common, traditionally separated information sources of software development get automatically extracted, related, and combined.

The ultimate goals of these techniques are to provide not only software engineers but also all other stakeholders better instruments to monitor, to comprehend, to discuss, and to steer software development activities. In particular, we will investigate how "software maps" as cartography-oriented, general-purpose, powerful visual analytics instruments can be used to improve software development effectiveness and transparency.

As precondition, our interested learners for this course shall have general knowledge about software development processes and procedures and have experience in IT-systems development or software maintenance. This course is especially interesting for

- IT-project managers
- Software developers, software testers and software engineers
- Software architects and modelers
- Parties responsible for financing the IT-development in a company

Content

Visual Software Analytics

Experiment 1 .

Objective

Getting familiar with the Software Cost Estimation Models.

Theory

The algorithmic method is designed to provide some mathematical equations to perform software estimation. These mathematical equations are based on research and historical data and use inputs such as Source Lines of Code (SLOC), number of functions to perform, and other cost drivers such as language, design methodology, skill-levels, risk assessments, etc. The algorithmic methods have been largely studied and there are a lot of models have been developed, such as COCOMO models, Putnam model, and function points based models.

General advantages:

1. It is able to generate repeatable estimations.
2. It is easy to modify input data, refine and customize formulas.
3. It is efficient and able to support a family of estimations or a sensitivity analysis.
4. It is objectively calibrated to previous experience.

General disadvantages:

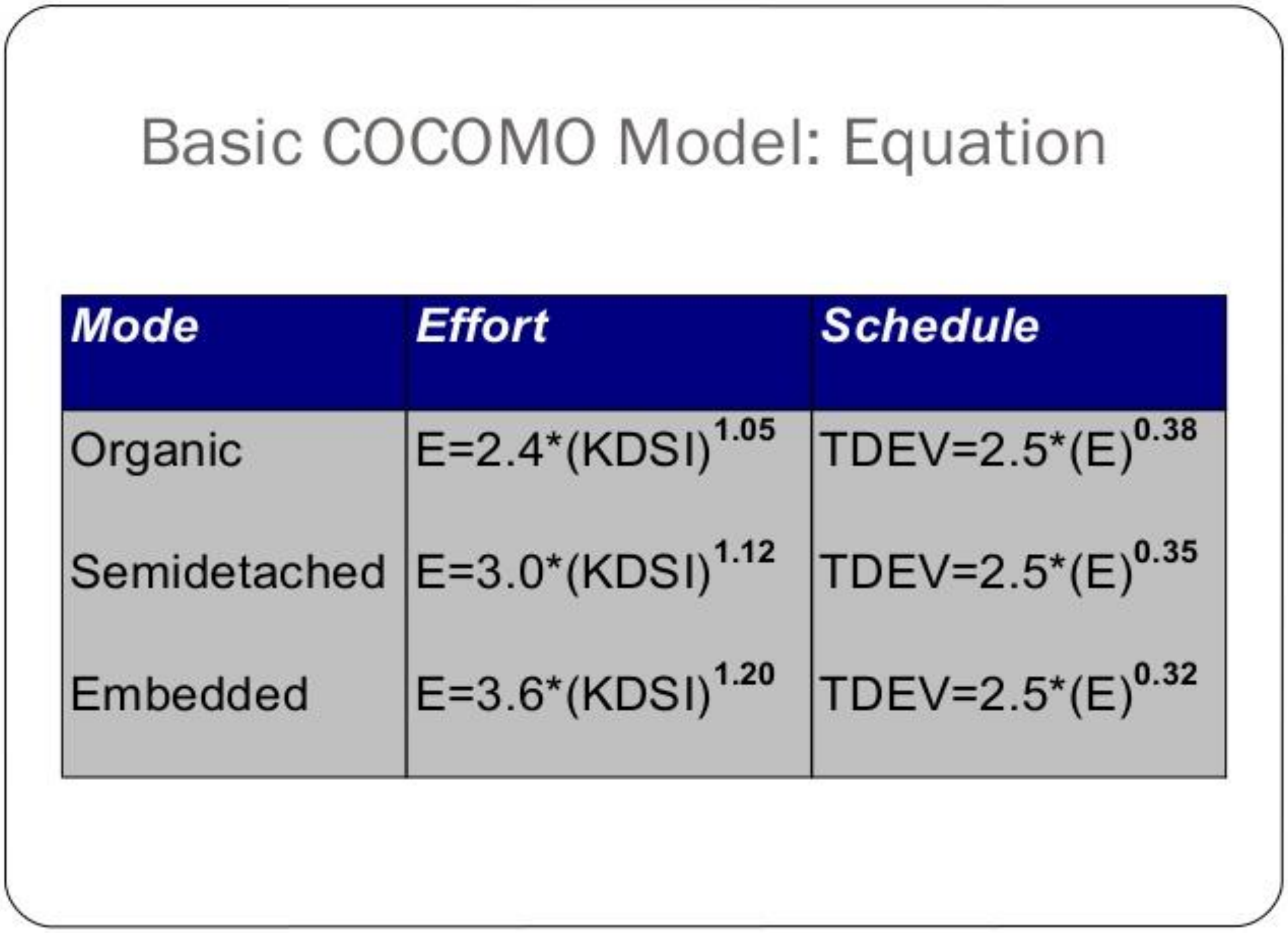
1. It is unable to deal with exceptional conditions, such as exceptional personnel in any software cost estimating exercises, exceptional teamwork, and an exceptional match between skill-levels and tasks.
2. Poor sizing inputs and inaccurate cost driver rating will result in inaccurate estimation.
3. Some experience and factors cannot be easily quantified.

COCOMO Models

One very widely used algorithmic software cost model is the Constructive Cost Model (COCOMO). The basic COCOMO model has a very simple form:

MAN-MONTHS = K1* (Thousands of Delivered Source Instructions) ^{K2}

Where K1 and K2 are two parameters dependent on the application and development environment.



Basic COCOMO Model: Equation

Mode	Effort	Schedule
Organic	$E=2.4*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semidetached	$E=3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=3.6*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

Estimates from the basic COCOMO model can be made more accurate by taking into account other factors concerning the required characteristics of the software to be developed, the qualification and experience of the development team, and the software development environment. Some of these factors are:

Complexity of the software

1. Required reliability
2. Size of data base
3. Required efficiency (memory and execution time)
4. Analyst and programmer capability
5. Experience of team in the application area
6. Experience of team with the programming language and computer
7. Use of tools and software engineering practices

Many of these factors affect the person months required by an order of magnitude or more. COCOMO assumes that the system and software requirements have already been defined, and that these requirements are stable. This is often not the case.

COCOMO model is a regression model. It is based on the analysis of 63 selected projects. The primary input is KDSI. The problems are:

1. In early phase of system life-cycle, the size is estimated with great uncertainty value. So, the accurate cost estimate cannot be arrived at.
2. The cost estimation equation is derived from the analysis of 63 selected projects. It usually has some problems outside of its particular environment. For this reason, the recalibration is necessary.

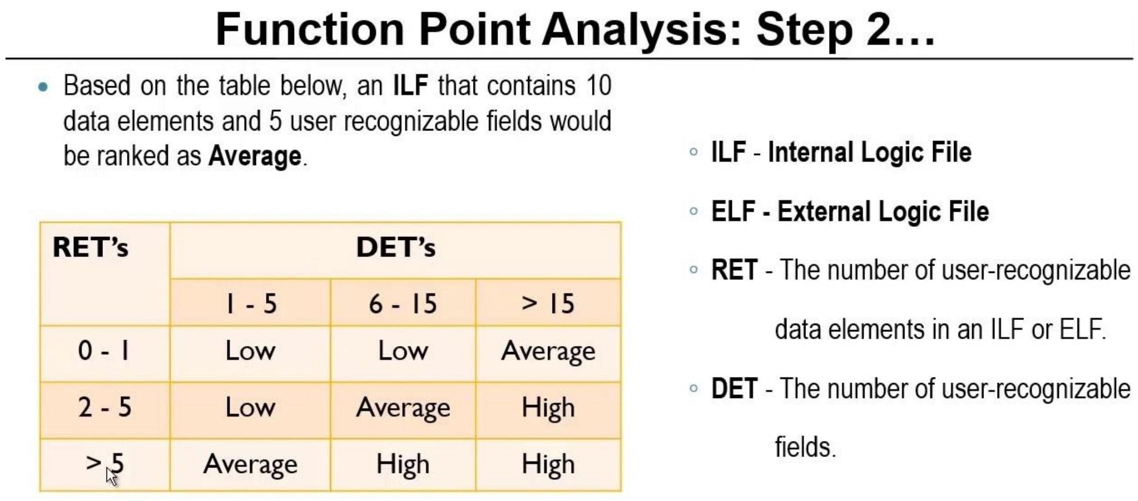
According to Kemerer's research, the average error for all versions of the model is 601%. The detailed model and Intermediate model seem not much better than basic model.

The first version of COCOMO model was originally developed in 1981. Now, it has been experiencing increasing difficulties in estimating the cost of software developed to new life cycle processes and capabilities including rapid-development process model, reuse-driven approaches, object-oriented approaches and software process maturity initiative.

For these reasons, the newest version, COCOMO 2.0, was developed. The major new modeling capabilities of COCOMO 2.0 are a tailorable family of software size models, involving object points, function points and source lines of code; nonlinear models for software reuse and reengineering; an exponent-driver approach for modeling relative software diseconomies of scale; and several additions, deletions, and updates to previous COCOMO effort-multiplier cost drivers. This new model is also serving as a framework for an extensive current data collection and analysis effort to further refine and calibrate the model's estimation capabilities.

Function Point Analysis Based Methods

From above two algorithmic models, we found they require the estimators to estimate the number of SLOC in order to get man-months and duration estimates. The Function Point Analysis is another method of quantifying the size and complexity of a software system in terms of the functions that the systems deliver to the user. A number of proprietary models for cost estimation have adopted a function point type of approach, such as ESTIMACS and SPQR/20.



Function Point Analysis: Step 2…

- Based on the table below, an **ILF** that contains 10 data elements and 5 user recognizable fields would be ranked as **Average**.

RET's	DET's		
	1 - 5	6 - 15	> 15
0 - 1	Low	Low	Average
2 - 5	Low	Average	High
> 5	Average	High	High

- **ILF** - **Internal Logic File**
- **ELF - External Logic File**
- **RET** - The number of user-recognizable data elements in an ILF or ELF.
- **DET** - The number of user-recognizable fields.

The function point measurement method was developed by Allan Albrecht at IBM and published in 1979. He believes function points offer several significant advantages over SLOC counts of size measurement. There are two steps in counting function points:

 Counting the user functions. The raw function counts are arrived at by considering a linear combination of five basic software components: external inputs, external outputs, external inquiries, logic internal files, and external interfaces, each at one of three complexity levels: simple, average or complex. The sum of these numbers, weighted according to the complexity level, is the number of function counts (FC).
 Adjusting for environmental processing complexity. The final function points are arrived at by multiplying FC by an adjustment factor that is determined by considering 14 aspects of processing complexity. This adjustment factor allows the FC to be modified by at most 35% or -35%.

The collection of function point data has two primary motivations. One is the desire by managers to monitor levels of productivity. Another use of it is in the estimation of software development cost.

There are some cost estimation methods which are based on a function point type of measurement, such as ESTIMACS and SPQR/20. SPQR/20 is based on a modified function point method. Whereas traditional function point analysis is based on evaluating 14 factors, SPQR/20 separates complexity into three categories: complexity of algorithms, complexity of code, and complexity of data structures. ESTIMACS is a propriety system designed to give development cost estimate at the conception stage of a project and it contains a module which estimates function point as a primary input for estimating cost.

The advantages of function point analysis based model are:

1. function points can be estimated from requirements specifications or design specifications, thus making it possible to estimate development cost in the early phases of development.
2. function points are independent of the language, tools, or methodologies used for implementation.
3. non-technical users have a better understanding of what function points are measuring since function points are based on the system user's external view of the system

From Kemerer's research, the mean error percentage of ESTIMACS is only 85.48%. So, considering the 601% with COCOMO and 771% with SLIM, I think the Function Point based cost estimation methods is the better approach especially in the early phases of development.

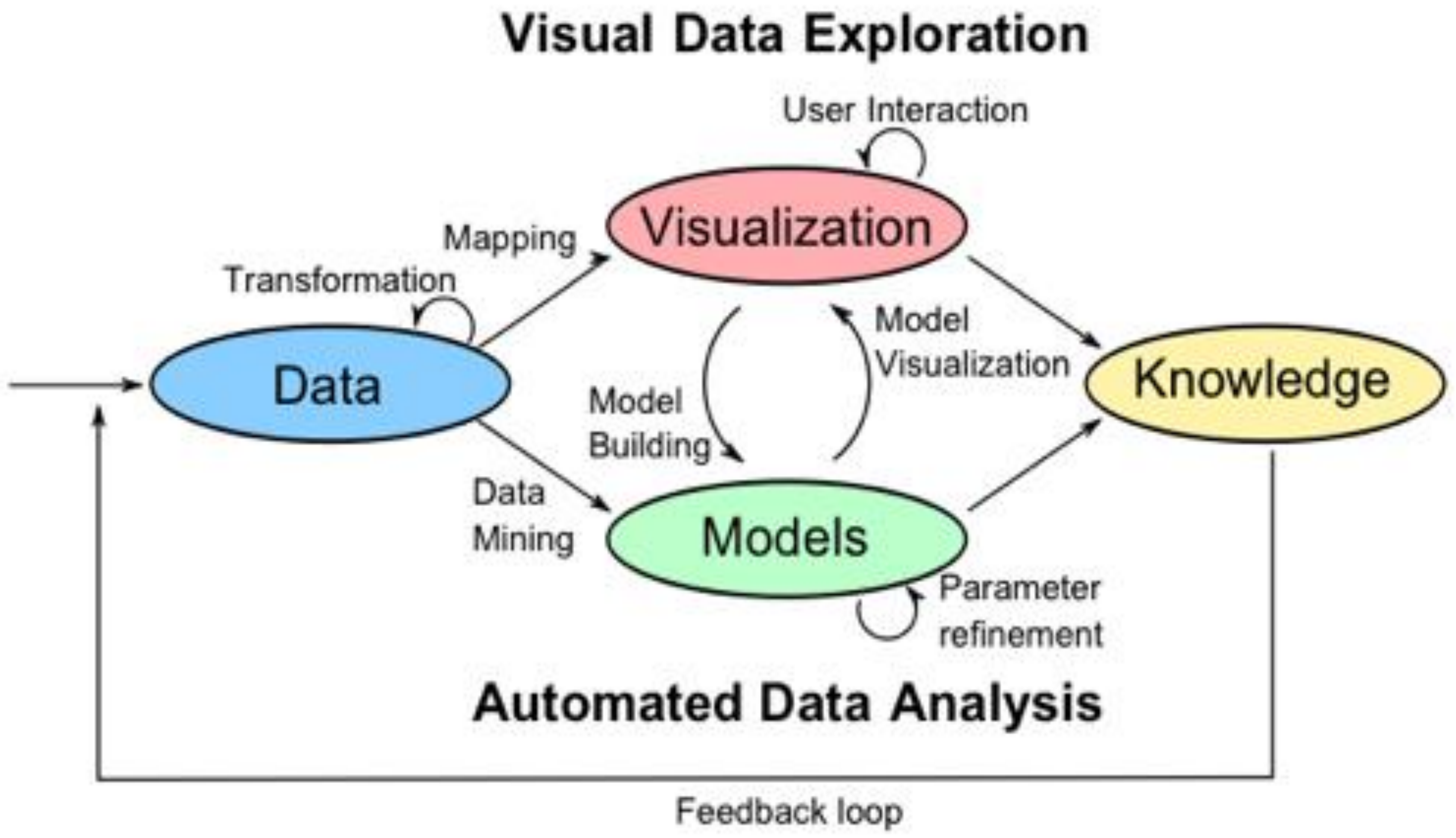
Experiment 2 .

Objective

Introduction to Visual Analytics

Theory

Visual Analytics is the science of analytical reasoning supported by interactive visual interfaces. Today, data is produced at an incredible rate and the ability to collect and store the data is increasing at a faster rate than the ability to analyze it. Over the last decades, a large number of automatic data analysis methods have been developed. However, the complex nature of many problems makes it indispensable to include human intelligence at an early stage in the data analysis process. Visual Analytics methods allow decision makers to combine their human flexibility, creativity, and background knowledge with the enormous storage and processing capacities of today's computers to gain insight into complex problems. Using advanced visual interfaces, humans may directly interact with the data analysis capabilities of today's computer, allowing them to make well-informed decisions in complex situations.



The first step is often to preprocess and transform the data to derive different representations for further exploration (as indicated by the Transformation arrow in the figure). Other typical preprocessing tasks include data cleaning, normalization, grouping, or integration of heterogeneous data sources.

After the transformation, the analyst may choose between applying visual or automatic analysis methods. If an automated analysis is used first, data mining methods are applied to generate models of the original data. Once a model is created the analyst has to evaluate and refine the models, which can best be done by interacting with the data. Visualizations allow the analysts to interact with the automatic methods by modifying parameters or selecting other analysis algorithms. Model visualization can then be used to evaluate the findings of the generated models. Alternating between visual and automatic methods is characteristic for the Visual Analytics process and leads to a continuous refinement and verification of preliminary results. Misleading results in an intermediate step can thus be discovered at an early stage, leading to better results and a higher confidence. If a visual data exploration is performed first, the user has to confirm the generated hypotheses by an automated analysis. User interaction with the visualization is needed to reveal insightful information, for instance by zooming in on different data areas or by considering different visual views on the data. Findings in the visualizations can be used to steer model building in the automatic analysis. In summary, in the Visual Analytics Process knowledge can be gained from visualization, automatic analysis, as well as the preceding interactions between visualizations, models, and the human analysts. the first step is often to preprocess and transform the data to derive different representations for further exploration (as indicated by the Transformation arrow in the figure). Other typical preprocessing tasks include data cleaning, normalization, grouping, or integration of heterogeneous data sources.

Experiment 3

Objective

Getting started with Python studying variables, user defined function, indentation.

Theory

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

An excellent way to see how Python code works is to run the Python interpreter and type code right into it. If you ever have a question like, "What happens if I add an int to a list?" Just typing it into the Python interpreter is a fast and likely the best way to see what happens. (See below to see what really happens!)

```
$ python         ## Run the Python interpreter
Python 2.7.9 (default, Dec 30 2014, 03:41:42)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-55)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 6        ## set a variable in this interpreter session
>>> a            ## entering an expression prints its value
6
>>> a + 2
8
>>> a = 'hi'     ## 'a' can hold a string just as well
>>> a
'hi'
>>> len(a)       ## call the len() function on a string
2
>>> a + len(a)   ## try something that doesn't work
Traceback (most recent call last):
  File "", line 1, in
TypeError: cannot concatenate 'str' and 'int' objects
>>> a + str(len(a))  ## probably what you really wanted
'hi2'
>>> foo          ## try something else that doesn't work
Traceback (most recent call last):
  File "", line 1, in
NameError: name 'foo' is not defined
>>> ^D           ## type CTRL-d to exit (CTRL-z in Windows/DOS terminal)
```

User-defined Functions

Functions in Python are defined like this

```
# Defines a "repeat" function that takes 2 arguments.
def repeat(s, exclaim):
    """
    Returns the string 's' repeated 3 times.
    If exclaim is true, add exclamation marks.
    """

    result = s + s + s # can also use "s * 3" which is faster (Why?)
    if exclaim:
        result = result + '!!!'
    return result
```

Notice also how the lines that make up the function or if-statement are grouped by all having the same level of indentation. We also presented 2 different ways to repeat strings, using the + operator which is more user-friendly, but * also works because it's Python's "repeat" operator, meaning that `'-' * 10` gives `'----------'`, a neat way to create an onscreen "line." In the code comment, we hinted that * works faster than +, the reason being that * calculates the size of the resulting object once whereas with +, that calculation is made each time + is called. Both + and * are called "overloaded" operators because they mean different things for numbers vs. for strings (and other data types).

The `def` keyword defines the function with its parameters within parentheses and its code indented. The first line of a function can be a documentation string ("docstring") that describes what the function does. The docstring can be a single line, or a multi-line description as in the example above. (Yes, those are "triple quotes," a feature unique to Python!) Variables defined in the function are local to that function, so the "result" in the above function is separate from a "result" variable in another function. The `return` statement can take an argument, in which case that is the value returned to the caller.

Indentation

One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

Python's use of whitespace feels a little strange at first, but it's logical and I found I got used to it very quickly. Avoid using TABs as they greatly complicate the indentation scheme (not to mention TABs may mean different things on different platforms). Set your editor to insert spaces instead of TABs for Python code.

```
def main():
    if name == 'Guido':
        print repeeeet(name) + '!!!'
    else:
        print repeat(name)
```

Experiment 4

Objective

Getting familiar with Python Dataframe and Visualization libraries such as matplot lib

Theory

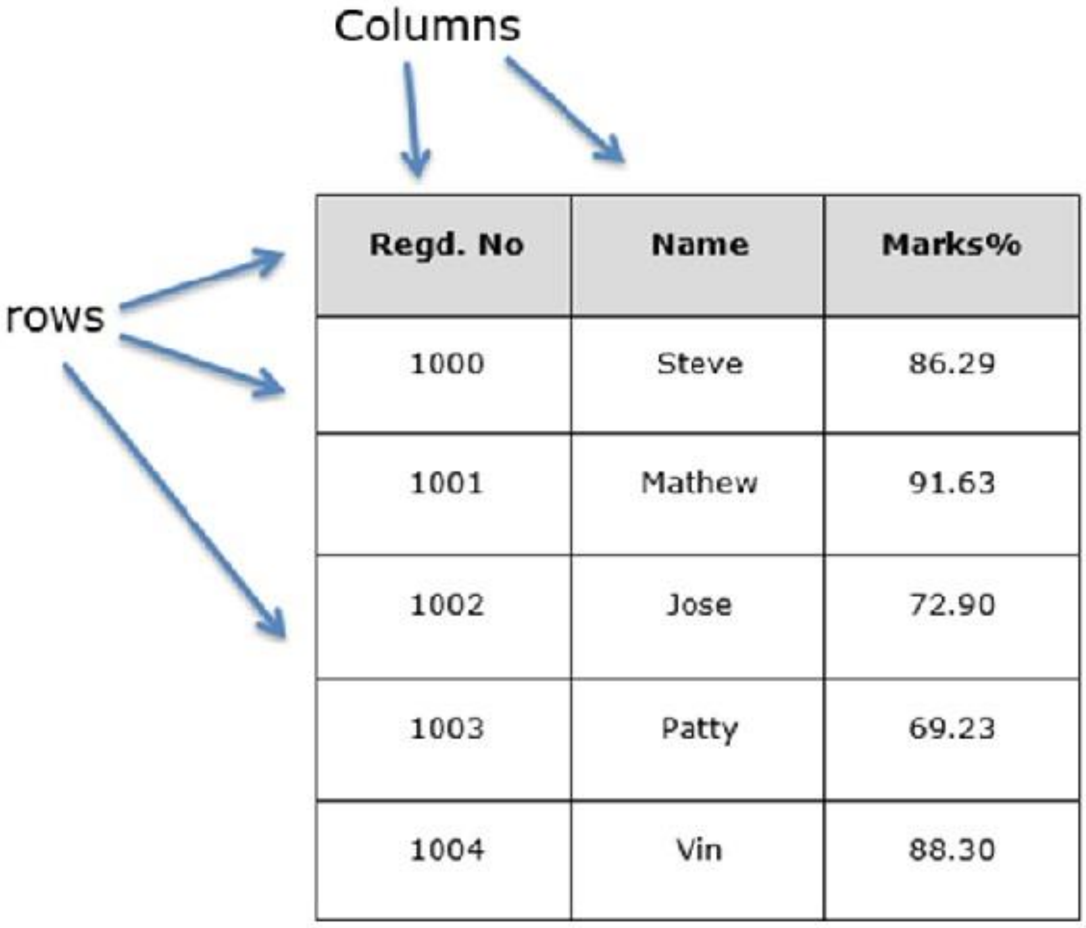
A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Features of DataFrame

- Potentially columns are of different types
- Size – Mutable
- Labeled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

Structure

Let us assume that we are creating a data frame with student's data.



Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30

You can think of it as an SQL table or a spreadsheet data representation.

pandas.DataFrame

A pandas DataFrame can be created using the following constructor

```
pandas.DataFrame( data, index, columns, dtype, copy)
```

The parameters of the constructor are as follows

S.No	Parameter & Description
1	**data** data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.
2	**index** For the row labels, the Index to be used for the resulting frame is Optional Default np.arrange(n) if no index is passed.
3	**columns** For column labels, the optional default syntax is - np.arrange(n). This is only true if no index is passed.
4	**dtype** Data type of each column.
4	**copy** This command (or whatever it is) is used for copying of data, if the default is False.

Create DataFrame

A pandas DataFrame can be created using various inputs like −

- Lists
- dict
- Series
- Numpy ndarrays
- Another DataFrame

In the subsequent sections of this chapter, we will see how to create a DataFrame using these inputs.

Create an Empty DataFrame

A basic DataFrame, which can be created is an Empty Dataframe.

Example

```
#import the pandas library and aliasing as pd

import pandas as pd

df = pd.DataFrame()

print df
```

Its **output** is as follows:

```
Empty DataFrame
Columns: []
Index: []
```

Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists.

Example 1

```
import pandas as pd

data = [1,2,3,4,5]

df = pd.DataFrame(data)

print df
```

Its **output** is as follows:

```
     0
0    1
1    2
2    3
3    4
4    5
```

Plot column values as a bar plot

```
import matplotlib.pyplot as plt

import pandas as pd



# a simple line plot

df.plot(kind='bar',x='name',y='age')
```

Experiment 5

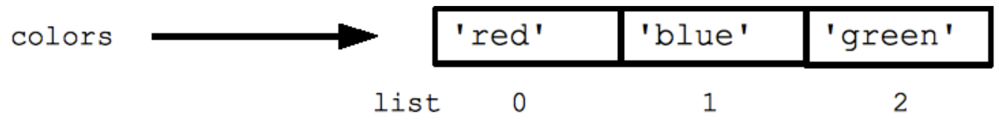
Objective

Working with List and Dictionaries in Python to handle data.

Theory

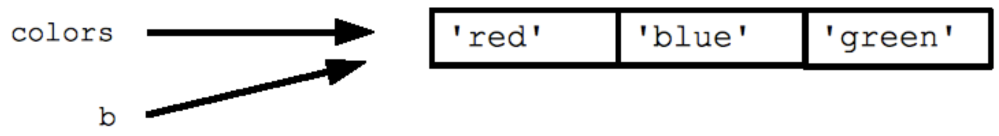
Python has a great built-in list type named "list". List literals are written within square brackets [ ]. Lists work similarly to strings -- use the len() function and square brackets [ ] to access data, with the first element at index 0.

```
colors = ['red', 'blue', 'green']
print colors[0]    ## red
print colors[2]    ## green
print len(colors)  ## 3
```



Assignment with an = on lists does not make a copy. Instead, assignment makes the two variables point to the one list in memory.

```
b = colors   ## Does not copy the list
```



The "empty list" is just an empty pair of brackets [ ]. The '+' works to append two lists, so [1, 2] + [3, 4] yields [1, 2, 3, 4] (this is just like + with strings).

FOR and IN

Python's *for* and *in* constructs are extremely useful, and the first use of them we'll see is with lists. The *for* construct -- for **var** in **list** -- is an easy way to look at each element in a list (or other collection). Do not add or remove from the list during iteration.

```
  squares = [1, 4, 9, 16]
  sum = 0
  for num in squares:
    sum += num
  print sum  ## 30
```

If you know what sort of thing is in the list, use a variable name in the loop that captures that information such as "num", or "name", or "url". Since python code does not have other syntax to remind you of types, your variable names are a key way for you to keep straight what is going on.

The *in* construct on its own is an easy way to test if an element appears in a list (or other collection) -- **value** in **collection** -- tests if the value is in the collection, returning True/False.

```
  list = ['larry', 'curly', 'moe']
  if 'curly' in list:
    print 'yay'
```

The for/in constructs are very commonly used in Python code and work on data types other than list, so you should just memorize their syntax. You may have habits from other languages where you start manually iterating over a collection, where in Python you should just use for/in.

You can also use for/in to work on a string. The string acts like a list of its chars, so for ch in s: print ch prints all the chars in a string.

Range

The range(n) function yields the numbers 0, 1, ... n-1, and range(a, b) returns a, a+1, ... b-1 -- up to but not including the last number. The combination of the for-loop and the range() function allow you to build a traditional numeric for loop:

```
  ## print the numbers from 0 through 99
  for i in range(100):
```

```
  print i
```

There is a variant xrange() which avoids the cost of building the whole list for performance sensitive cases (in Python 3000, range() will have the good performance behavior and you can forget about xrange()).

While Loop

Python also has the standard while-loop, and the *break* and *continue* statements work as in C++ and Java, altering the course of the innermost loop. The above for/in loops solves the common case of iterating over every element in a list, but the while loop gives you total control over the index numbers. Here's a while loop which accesses every 3rd element in a list:

```
## Access every 3rd element in a list
i = 0
while i < len(a):
  print a[i]
  i = i + 3
```

List Methods

Here are some other common list methods.

- list.append(elem) -- adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.
- list.insert(index, elem) -- inserts the element at the given index, shifting elements to the right.
- list.extend(list2) adds the elements in list2 to the end of the list. Using + or += on a list is similar to using extend().
- list.index(elem) -- searches for the given element from the start of the list and returns its index. Throws a ValueError if the element does not appear (use "in" to check without a ValueError).
- list.remove(elem) -- searches for the first instance of the given element and removes it (throws ValueError if not present)
- list.sort() -- sorts the list in place (does not return it). (The sorted() function shown later is preferred.)
- list.reverse() -- reverses the list in place (does not return it)
- list.pop(index) -- removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of append()).

Notice that these are *methods* on a list object, while len() is a function that takes the list (or string or whatever) as an argument.

```
list = ['larry', 'curly', 'moe']
list.append('shemp')         ## append elem at end
list.insert(0, 'xxx')        ## insert elem at index 0
list.extend(['yyy', 'zzz'])  ## add list of elems at end
print list  ## ['xxx', 'larry', 'curly', 'moe', 'shemp', 'yyy', 'zzz']
print list.index('curly')    ## 2

```

```
  list.remove('curly')          ## search and remove that element
  list.pop(1)                   ## removes and returns 'larry'
  print list  ## ['xxx', 'moe', 'shemp', 'yyy', 'zzz']
```

Common error: note that the above methods do not *return* the modified list, they just modify the original list.

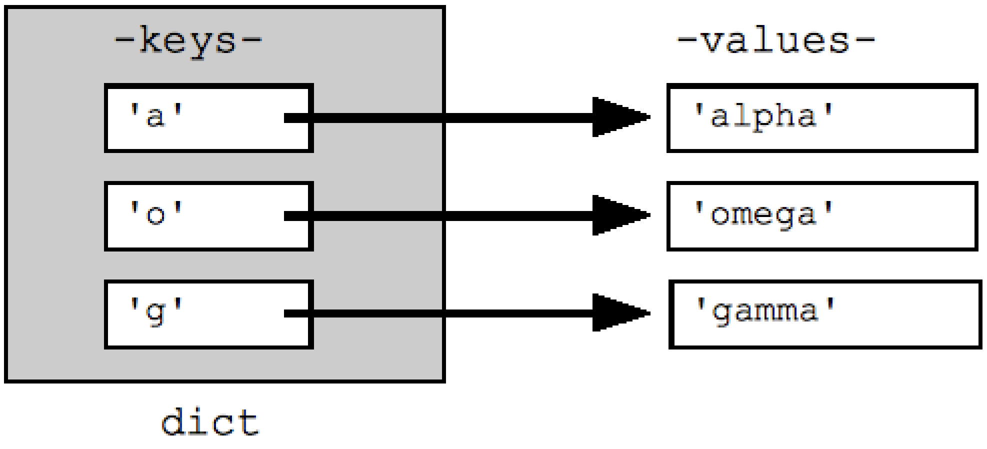
```
  list = [1, 2, 3]
  print list.append(4)   ## NO, does not work, append() returns None
  ## Correct pattern:
  list.append(4)
  print list  ## [1, 2, 3,4]
```

Dict Hash Table

Python's efficient key/value hash table structure is called a "dict". The contents of a dict can be written as a series of key:value pairs within braces { }, e.g. dict = {key1:value1, key2:value2, ... }. The "empty dict" is just an empty pair of curly braces {}.

Looking up or setting a value in a dict uses square brackets, e.g. dict['foo'] looks up the value under the key 'foo'. Strings, numbers, and tuples work as keys, and any type can be a value. Other types may or may not work correctly as keys (strings and tuples work cleanly since they are immutable). Looking up a value which is not in the dict throws a KeyError -- use "in" to check if the key is in the dict, or use dict.get(key) which returns the value or None if the key is not present (or get(key, not-found) allows you to specify what value to return in the not-found case).

```
## Can build up a dict by starting with the the empty dict {}
## and storing key/value pairs into the dict like this:
## dict[key] = value-for-that-key
dict = {}
dict['a'] = 'alpha'
dict['g'] = 'gamma'
dict['o'] = 'omega'

print dict  ## {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}

print dict['a']     ## Simple lookup, returns 'alpha'
dict['a'] = 6       ## Put new key/value into dict
'a' in dict         ## True
## print dict['z']                  ## Throws KeyError
if 'z' in dict: print dict['z']     ## Avoid KeyError
print dict.get('z')  ## None (instead of KeyError)
```



A for loop on a dictionary iterates over its keys by default. The keys will appear in an arbitrary order. The methods dict.keys() and dict.values() return lists of the keys or values explicitly. There's also an items() which returns a list of (key, value) tuples, which is the most efficient way to examine all the key value data in the dictionary. All of these lists can be passed to the sorted() function.

```
  ## By default, iterating over a dict iterates over its keys.
  ## Note that the keys are in a random order.
  for key in dict: print key
  ## prints a g o

  ## Exactly the same as above
  for key in dict.keys(): print key

  ## Get the .keys() list:
  print dict.keys()  ## ['a', 'o', 'g']

  ## Likewise, there's a .values() list of values
  print dict.values()  ## ['alpha', 'omega', 'gamma']

  ## Common case -- loop over the keys in sorted order,
  ## accessing each key/value
  for key in sorted(dict.keys()):
    print key, dict[key]

  ## .items() is the dict expressed as (key, value) tuples
  print dict.items()  ##  [('a', 'alpha'), ('o', 'omega'), ('g', 'gamma')]

  ## This loop syntax accesses the whole dict by looping
  ## over the .items() tuple list, accessing one (key, value)
  ## pair on each iteration.
  for k, v in dict.items(): print k, '>', v
  ## a > alpha    o > omega     g > gamma
```

There are "iter" variants of these methods called iterkeys(), itervalues() and iteritems() which avoid the cost of constructing the whole list -- a performance win if the data is huge. However, I generally prefer the plain keys() and values() methods with their sensible names. In Python 3000 revision, the need for the iterkeys() variants is going away.

Strategy note: from a performance point of view, the dictionary is one of your greatest tools, and you should use it where you can as an easy way to organize data. For example, you might read a log file where each line begins with an IP address, and store the data into a dict using the IP address as the key, and the list of lines where it appears as the value. Once you've read in the whole file, you can look up any IP address and instantly see its list of lines. The dictionary takes in scattered data and makes it into something coherent

Experiment 6

Objective

Code Analyzer Tool for analyzing line of code, number of packages, number of comments.

Theory

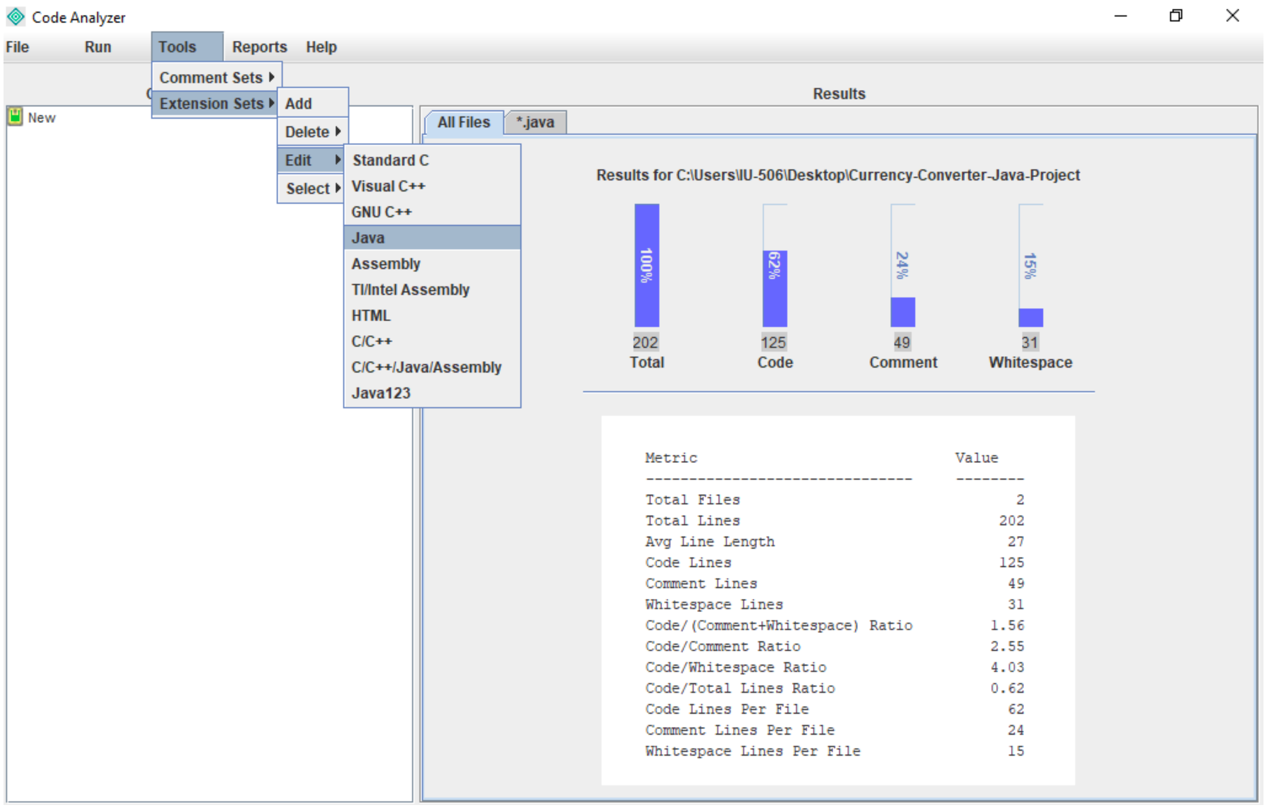
Lines of Code

The phrase “lines of code” (**LOC**) is a metric generally used to evaluate a **software** program or codebase according to its size. It is a general identifier taken by adding up the number of lines of code used to write a program.

White Spaces

In computer programming, whitespace is any character or series of characters that represent horizontal or vertical space in typography. When rendered, a whitespace character does not correspond to a visible mark, but typically does occupy an area on a page.

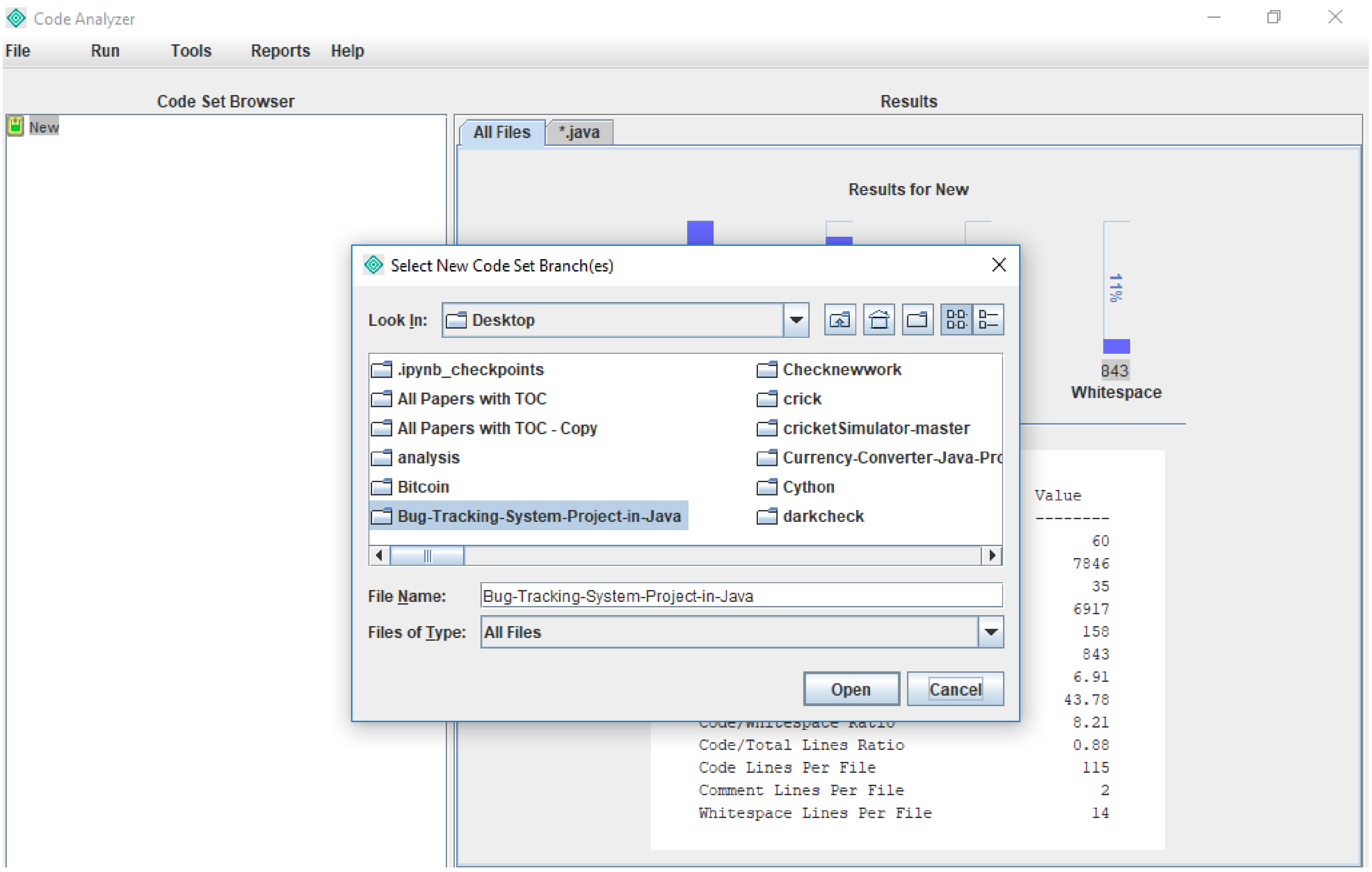
Select the extension of the program



Results for C:\Users\IU-506\Desktop\Currency-Converter-Java-Project

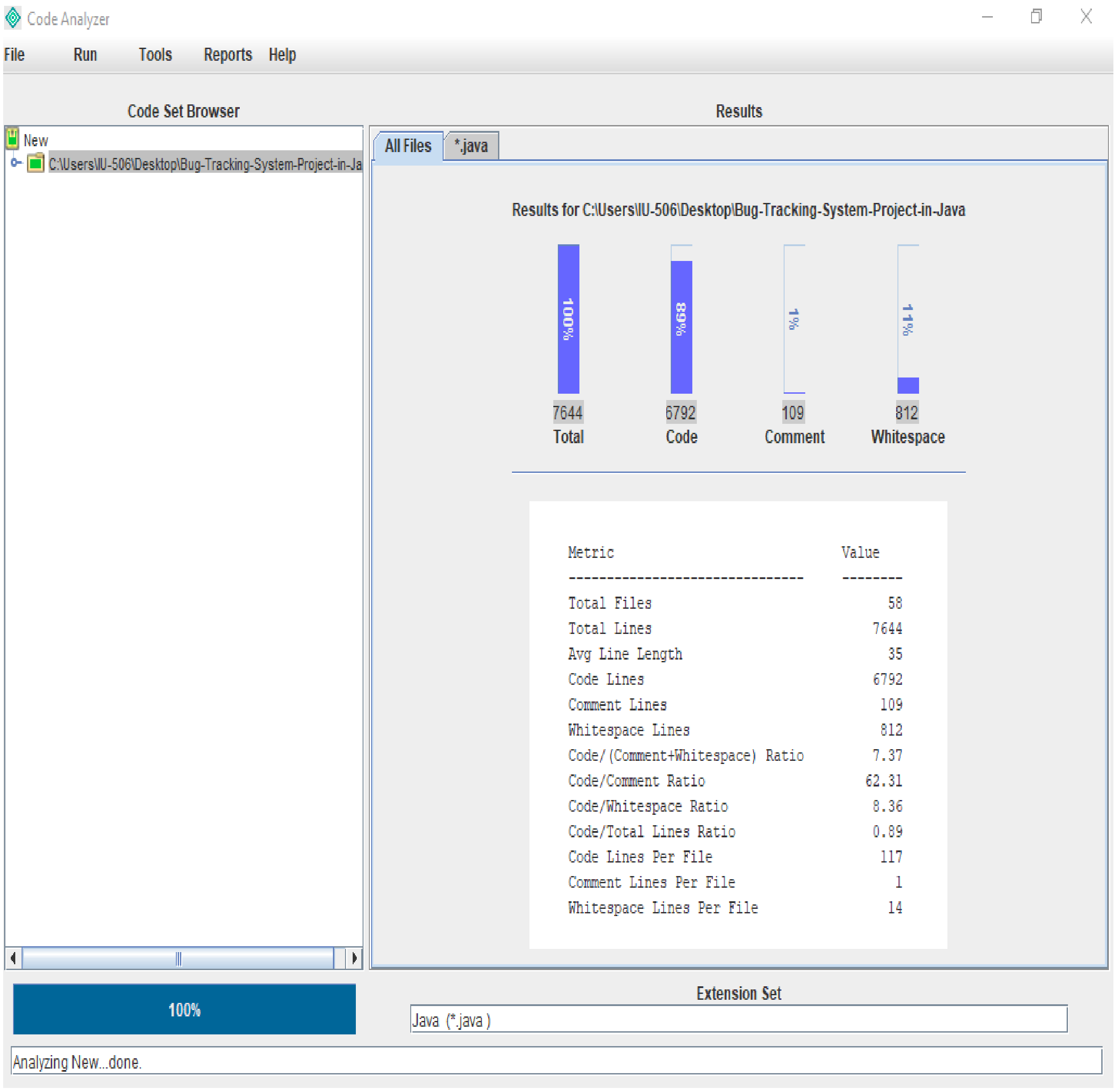
```
Metric                              Value
-------------------------------     --------
Total Files                                2
Total Lines                              202
Avg Line Length                           27
Code Lines                               125
Comment Lines                             49
Whitespace Lines                          31
Code/(Comment+Whitespace) Ratio         1.56
Code/Comment Ratio                      2.55
Code/Whitespace Ratio                   4.03
Code/Total Lines Ratio                  0.62
Code Lines Per File                       62
Comment Lines Per File                    24
Whitespace Lines Per File                 15
```

Add Project



File Name: Bug-Tracking-System-Project-in-Java

```
                                   Value
                                   --------
                                         60
                                       7846
                                         35
                                       6917
                                        158
                                        843
                                       6.91
                                      43.78
Code/Whitespace Ratio                  8.21
Code/Total Lines Ratio                 0.88
Code Lines Per File                     115
Comment Lines Per File                    2
Whitespace Lines Per File                14
```



```
Metric                            Value
-------------------------------   --------
Total Files                             58
Total Lines                           7644
Avg Line Length                         35
Code Lines                            6792
Comment Lines                          109
Whitespace Lines                       812
Code/(Comment+Whitespace) Ratio       7.37
Code/Comment Ratio                   62.31
Code/Whitespace Ratio                 8.36
Code/Total Lines Ratio                0.89
Code Lines Per File                    117
Comment Lines Per File                   1
Whitespace Lines Per File               14
```

Analyzing New...done.

Experiment 7

Objective

Program Halstead's metrics for finding Software Metrics.

Theory

A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands. Halstead's metrics are included in a number of current commercial tools that count software lines of code. By counting the tokens and determining which are operators and which are operands, the following base measures can be collected :

n1 = Number of distinct operators.
n2 = Number of distinct operands.
N1 = Total number of occurrences of operators.
N2 = Total number of occurrences of operands.

In addition to the above, Halstead defines the following :

n1* = Number of potential operators.
n2* = Number of potential operands.

Halstead refers to n1* and n2* as the minimum possible number of operators and operands for a module and a program respectively. This minimum number would be embodied in the programming language itself, in which the required operation would already exist (for example, in C language, any program must contain at least the definition of the function main()), possibly as a function or as a procedure: n1* = 2, since at least 2 operators must appear for any function or procedure : 1 for the name of the function and 1 to serve as an assignment or grouping symbol, and n2* represents the number of parameters, without repetition, which would need to be passed on to the function or the procedure.

Halstead metrics –
Halstead metrics are :

Halstead Program Length – The total number of operator occurrences and the total number of operand occurrences.
N = N1 + N2

And estimated program length is, N^ = n1log2n1 + n2log2

Halstead Vocabulary – The total number of unique operator and unique operand occurrences.
n = n1 + n2

Program Volume – Proportional to program size, represents the size, in bits, of space necessary for storing the program. This parameter is dependent on specific algorithm implementation. The properties V, N, and the number of lines in the code are shown to be linearly connected and equally valid for measuring relative program size.
V = Size * (log2 vocabulary) = N * log2(n)

The unit of measurement of volume is the common unit for size “bits”. It is the actual size of a program if a uniform binary encoding for the vocabulary is used. And error = Volume / 3000

Program Difficulty – This parameter shows how difficult to handle the program is.
D = (n1 / 2) * (N2 / n2)
D = 1 / L
As the volume of an implementation of a program increases, the program level decreases and the difficulty increases. Thus, programming practices such as redundant usage of operands, or the failure to use higher-level control constructs will tend to increase the volume as well as the difficulty.

```
import math
import time
from metricbase import MetricBase
from globals import *

class HalsteadMetric( MetricBase ):
    """ Compute various HalsteadMetric metrics. """
    totalV = 0
    totalE = 0
    numModules = 0
    def __init__( self, context, runMetrics, metrics, pa, *args, **kwds ):
        """ Initialization for the HalsteadMetric metrics."""
        self.inFile = context['inFile']
        self.context = context
        self.runMetrics = runMetrics
        self.metrics = metrics
        self.pa = pa
        self.inFile = context['inFile']
        self.numOperators = 0
        self.numOperands = 0
        self.uniqueOperators = {}
        self.uniqueOperands = {}
        HalsteadMetric.numModules += 1

        # initialize category accummulators as dictionaries
        self.hsDict = {}
        for t in ['token','stmt','block','function','class','module','run']:
            self.uniqueOperators[t] = {}
            self.uniqueOperands[t] = {}
            #for v in ['N','N1','N2','n','n1','n2','V','D','E','avgV','avgE']:
            #    self.hsDict[(t,v)] = 0
```

```
def processToken( self, currentFcn, currentClass, tok, *args, **kwds ):
    """ Collect token data for Halstead metrics."""
    if tok.type in [WS, EMPTY, ENDMARKER, NEWLINE, EMPTY, COMMENT]:
        pass
    elif tok.type in [OP, INDENT, DEDENT]:
        self.numOperators += 1
        self.uniqueOperators['token'][tok.text] = self.uniqueOperators['token'].get(tok.text, 0) + 1
    else:
        self.numOperands += 1
        sDict = self.context.__repr__()
        k = (sDict,tok.text)
        self.uniqueOperands['token'][k] = self.uniqueOperands['token'].get(tok.text, 0) + 1

def processStmt( self, currentFcn, currentClass, stmt, *args, **kwds ):
    """ Collect statement data for Halstead metrics."""

    result = None

    # the two lines following this comment would compute the Halstead
    # metrics for each statement in the run, However, it is
    # normally overkill, so these lines are commented out.

    #lineNum = stmt[0].row
    #result = self.computeCategory( 'stmt', lineNum, stmt )

    return result

def processBlock( self, currentFcn, currentClass, block, *args, **kwds ):
    """ Collect block data for Halstead metrics."""

    result = None

    # the two lines following this comment would compute the Halstead
    # metrics for each statement in the run, However, it is
    # normally overkill, so the two lines are commented out.

    #blockNum = self.context['blockNum']
    #result = self.computeCategory( 'block', blockNum, block )

    return result

def processFunction( self, currentFcn, currentClass, fcn, *args, **kwds ):
    """ Collect function data for Halstead metrics."""
    result = self.computeCategory( 'function', currentFcn, fcn )
    return result

def processClass( self, currentFcn, currentClass, cls, *args, **kwds ):
    """ Collect class data for Halstead metrics."""
    result = self.computeCategory( 'class', currentClass, cls )
```

```
        return result

    def processModule( self, moduleName, mod, *args, **kwds ):
        """ Collect module data for Halstead metrics."""
        result = self.computeCategory( 'module', moduleName, mod )
        return result

    def processRun( self, run, *args, **kwds ):
        """ Collect run data for Halstead metrics."""
        datestamp = time.strftime("%Y-%m-%d.%H:%m%Z",time.localtime())
        result = self.computeCategory( 'run', datestamp, run )
        return result

    def __LOGb( self, x, b ):
        """ convert to LOGb(x) from natural logs."""
        try:
            result = math.log( x ) / math.log ( b )
        except OverflowError:
            result = 1.0
        return result

    def computeIncr( self, cat, tok, uniqueOperators, uniqueOperands ):
        """ Compute increment for token depending on which category it falls into."""
        operatorIncr = operandIncr = 0
        if tok.type in [WS, EMPTY, ENDMARKER, NEWLINE, EMPTY, COMMENT]:
            return (operatorIncr,operandIncr)

        if tok.type in [OP, INDENT, DEDENT]:
            operatorIncr = 1
            uniqueOperators[tok.text] = uniqueOperators.get(tok.text, 0) + 1
        else:
            operandIncr = 1
            uniqueOperands[tok.text] = uniqueOperands.get(tok.text,0) + 1

        return (operatorIncr,operandIncr)

    def computeCategory( self, cat, mod, lst ):
        """ Collection data for cat of code."""
        modID= id( mod )
        numOperators = numOperands = 0
        for tok in lst:
            result = self.computeIncr( cat, tok, self.uniqueOperators[cat], self.uniqueOperands[cat] )
            numOperators += result[0]
            numOperands += result[1]
        result = self.compute( cat, modID, numOperators, numOperands, self.uniqueOperators[cat],
self.uniqueOperands[cat] )
        return result

    def compute( self, cat, modID, numOperators, numOperands, uniqueOperators, uniqueOperands,
*args, **kwds ):
        """ Do actual calculations here."""

        n1 = len( uniqueOperands )
        n2 = len( uniqueOperators )
        N1 = numOperands
        N2 = numOperators
        N = N1 + N2
        n = n1 + n2
        V = float(N) * self.__LOGb( n, 2 )
        try:
            D = (float(n1)/2.0) * (float(N2)/float(n2))
        except ZeroDivisionError:
            D = 0.0
        E = D * V
        HalsteadMetric.totalV += V
        HalsteadMetric.totalE += E
        avgV = HalsteadMetric.totalV / HalsteadMetric.numModules
        avgE = HalsteadMetric.totalE / HalsteadMetric.numModules

        self.hsDict[(cat,modID,'n1')] = n1
        self.hsDict[(cat,modID,'n2')] = n2
        self.hsDict[(cat,modID,'N1')] = N1
        self.hsDict[(cat,modID,'N2')] = N2
        self.hsDict[(cat,modID,'N')] = N
        self.hsDict[(cat,modID,'n')] = n
        self.hsDict[(cat,modID,'V')] = V
        self.hsDict[(cat,modID,'D')] = D
        self.hsDict[(cat,modID,'E')] = E
        self.hsDict[(cat,modID,'numModules')] = HalsteadMetric.numModules
        self.hsDict[(cat,modID,'avgV')] = avgV
        self.hsDict[(cat,modID,'avgE')] = avgE

        return self.hsDict

    def display( self, cat=None ):
        """ Display the computed Halstead Metrics."""
        if self.pa.quietSw:
            return self.hsDict

        hdr = "\nHalstead Metrics for %s" % self.inFile
        print hdr
        print "-"*len(hdr) + '\n'

        if len( self.hsDict ) == 0:
            print "%-8s %-30s " % ('**N/A**','All Halstead metrics are zero')
            return self.hsDict

        keyList = self.hsDict.keys()
        keyList.sort()
        if 0:
            for k,i,v in keyList:
                if cat:
                    if k!=cat:
                        continue
                print "%14.2f %s %s %s" % (self.hsDict[(k,i,v)],k,i,v)
            print
        hdr1 = "Category Identifier                          D        E     N   N1   N2        V     avgE     avgV     n
n1   n2"
        hdr2 = "-------- -------------------------------- -------- -------- ----- ---- ---- -------- -------- -------- ----- ---- ----"
        #      12345678 123456789012345678901234567890  12345678 12345678 12345 1234 1234
12345678 12345678 12345678 12345 1234 1234
        fmt1 = "%-8s %-33s "
        fmt2 = "%8.2e %8.2e %5d %4d %4d %8.2e %8.2e %8.2e %5d %4d %4d"

        # this loop uses the Main Line Standards break logic. It does this to convert the
        # normal vertical output to a horizontal format. The control variables are the
        # category name and the identifier value.

        oldK = oldI = None
        vDict = {}
        vList = []
        hdrSw = True                # output header for first time thru
        for k,i,v in keyList:
            # only print data for the category we want
            if cat:
                if k != cat:
                    continue

            if v == "numModules":    # ignore this value for now
                continue

            if (oldK,oldI) != (k,i):    # change in category/id
                if oldK and oldI:           # this is not first time thru
                    #t = tuple([self.hsDict[(k,i,v)] for v in vList])
                    t = tuple([vDict[v] for v in vList])
                    print fmt1 % (k,i),
                    print fmt2 % t
                # initialize for next set of category/id
                vDict = {}
                vDict[v] = self.hsDict[(k,i,v)]
                vList = []
                vList.append( v )
                oldK = k
                oldI = i
                if hdrSw:
                    print hdr1
                    print hdr2
                    hdrSw = False
            else:       # we are still in the same category/id
                vDict[v] = self.hsDict[(k,i,v)]
                vList.append( v )

        print

        return self.hsDict
```

Experiment 8

Objective

Tree maps for visualizing data.

Theory

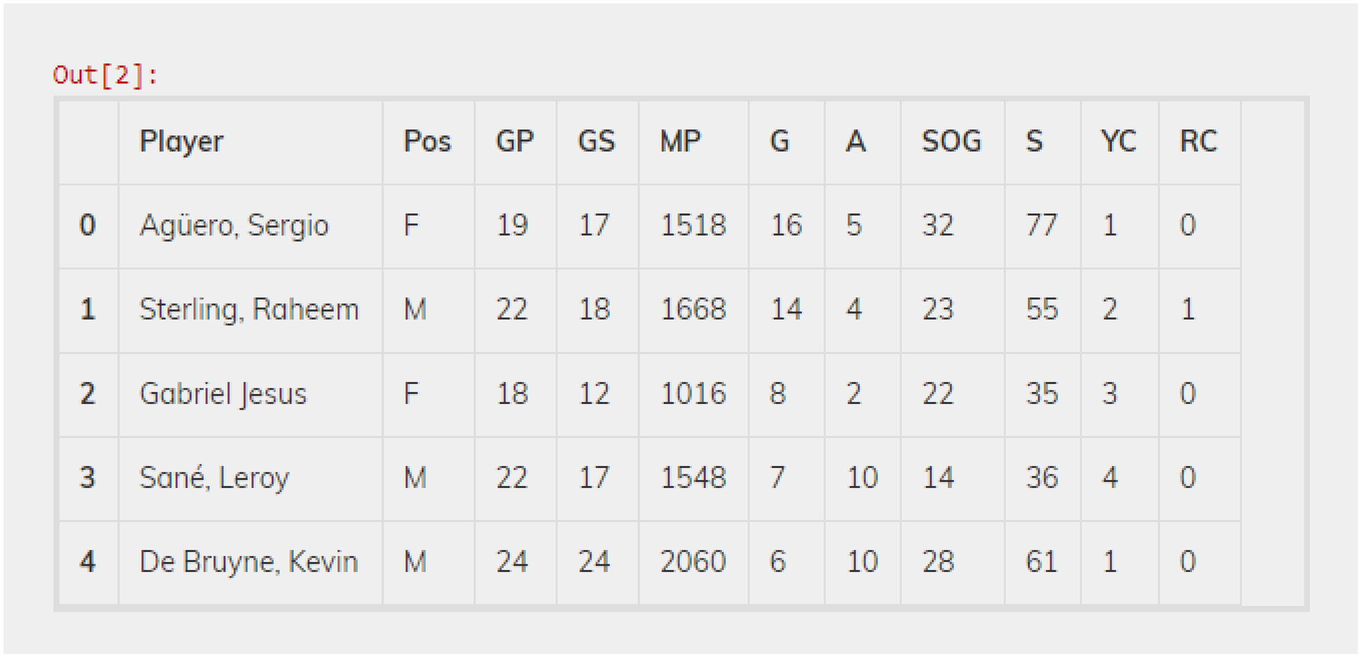
Python Treemaps with Squarify & Matplotlib

Treemaps are visualisations that split the area of our chart to display the value of our datapoints. At their simplest, they display shapes in sizes appropriate to their value, so bigger rectangles represent higher values. Python allows us to create these charts quite easily, as it will calculate the size of each rectangle for us and plot it in a way that fits. In addition to this, we can combine our treemap with the matplotlib library's ability to scale colours against variables to make good looking and easy to understand plots with Python.

Let's fire up our libraries (make sure you install squarify!) and take a look at our data:

```
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import squarify
```

```
data = pd.read_csv("Data/ManCityATT.csv")
data.head()
```



Out[2]:

	Player	Pos	GP	GS	MP	G	A	SOG	S	YC	RC
0	Agüero, Sergio	F	19	17	1518	16	5	32	77	1	0
1	Sterling, Raheem	M	22	18	1668	14	4	23	55	2	1
2	Gabriel Jesus	F	18	12	1016	8	2	22	35	3	0
3	Sané, Leroy	M	22	17	1548	7	10	14	36	4	0
4	De Bruyne, Kevin	M	24	24	2060	6	10	28	61	1	0

Our dataframe has a record for each player in Manchester City's squad, with their games/minutes played alongside goals, assists, shots and card data.Facing Manchester City next week, we would like to visualise where their threat comes from and who they rely on for goals and assists. We'll do this with our treemap!We will create our treemap in a few key steps:

Create a new dataframe that contains only players that have scored. Utilize matplotlib to create a colour map that assigns each player a colour according to how many goals they have scored.

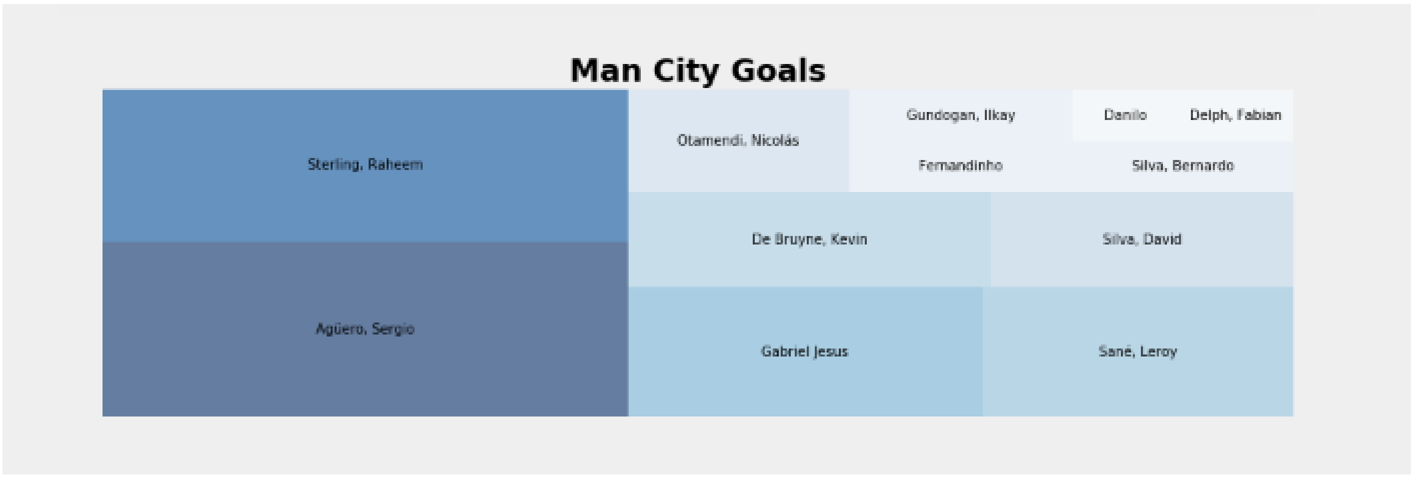
Set up a new, rectangular plot for our heatmap

Plot our data & title

Show the plot, with no axes

The commented code below will show you exactly how you can do this:

```
dataGoals = data[data["G"]>0]

#Utilise matplotlib to scale our goal numbers between the min and max, then assign this scale to our values.
norm = matplotlib.colors.Normalize(vmin=min(dataGoals.G), vmax=max(dataGoals.G))
colors = [matplotlib.cm.Blues(norm(value)) for value in dataGoals.G]

#Create our plot and resize it.
fig = plt.gcf()
ax = fig.add_subplot()
fig.set_size_inches(16, 4.5)

#Use squarify to plot our data, label it and add colours. We add an alpha layer to ensure black labels show through
squarify.plot(label=dataGoals.Player,sizes=dataGoals.G, color = colors, alpha=.6)
plt.title("Man City Goals",fontsize=23,fontweight="bold")

#Remove our axes and display the plot
plt.axis('off')
plt.show()
```



Experiment 9

Objective

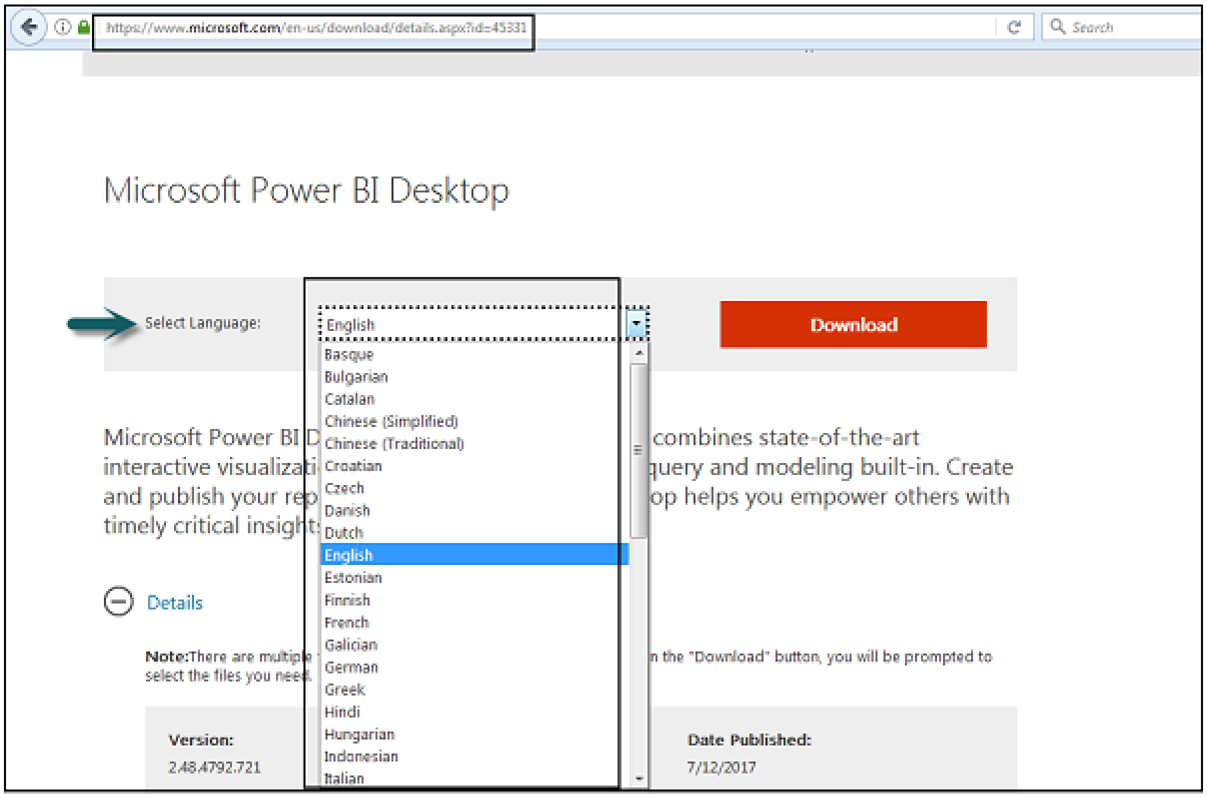
Getting Started with Power BI.

Theory

Power BI is a Data Visualization and Business Intelligence tool that converts data from different data sources to interactive dashboards and BI reports. Power BI suite provides multiple software, connector, and services - Power BI desktop, Power BI service based on SaaS, and mobile Power BI apps available for different platforms. These set of services are used by business users to consume data and build BI reports.

Power BI desktop app is used to create reports, while Power BI Services (Software as a Service - SaaS) is used to publish the reports, and Power BI mobile app is used to view the reports and dashboards.

Users can select a language in which they want to install Power BI and following files are available for download.



https://www.microsoft.com/en-us/download/details.aspx?id=45331

Microsoft Power BI Desktop

Select Language: English

Microsoft Power BI D… combines state-of-the-art interactive visualizati… query and modeling built-in. Create and publish your rep… op helps you empower others with timely critical insight…

Details

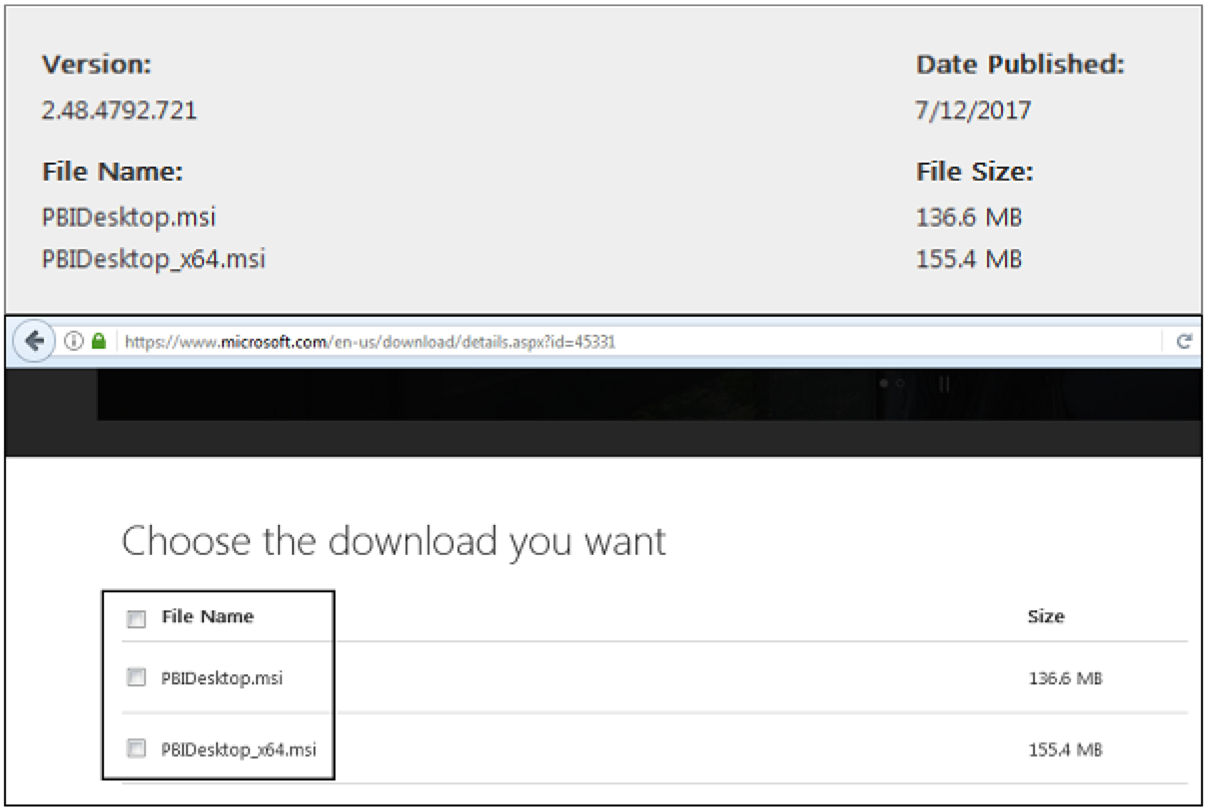
Note: There are multiple … n the "Download" button, you will be prompted to select the files you need.

Version: 2.48.4792.721

Date Published: 7/12/2017

This is the link to directly download Power BI files –

https://www.microsoft.com/en-us/download/details.aspx?id=45331



Version:	Date Published:
2.48.4792.721	7/12/2017

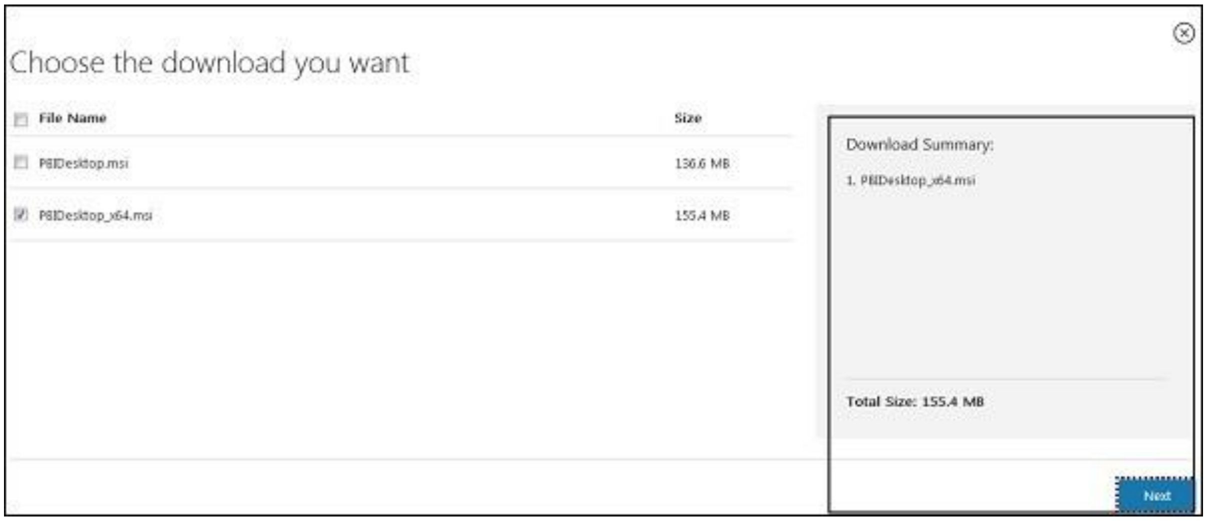
File Name:	File Size:
PBIDesktop.msi	136.6 MB
PBIDesktop_x64.msi	155.4 MB

https://www.microsoft.com/en-us/download/details.aspx?id=45331

Choose the download you want

File Name	Size
PBIDesktop.msi	136.6 MB
PBIDesktop_x64.msi	155.4 MB

PBIDesktop_x64.msi shows a 64-bit OS file. Select the file you want to install as per OS type and click Next. Save the installation file on the local drive.



Choose the download you want

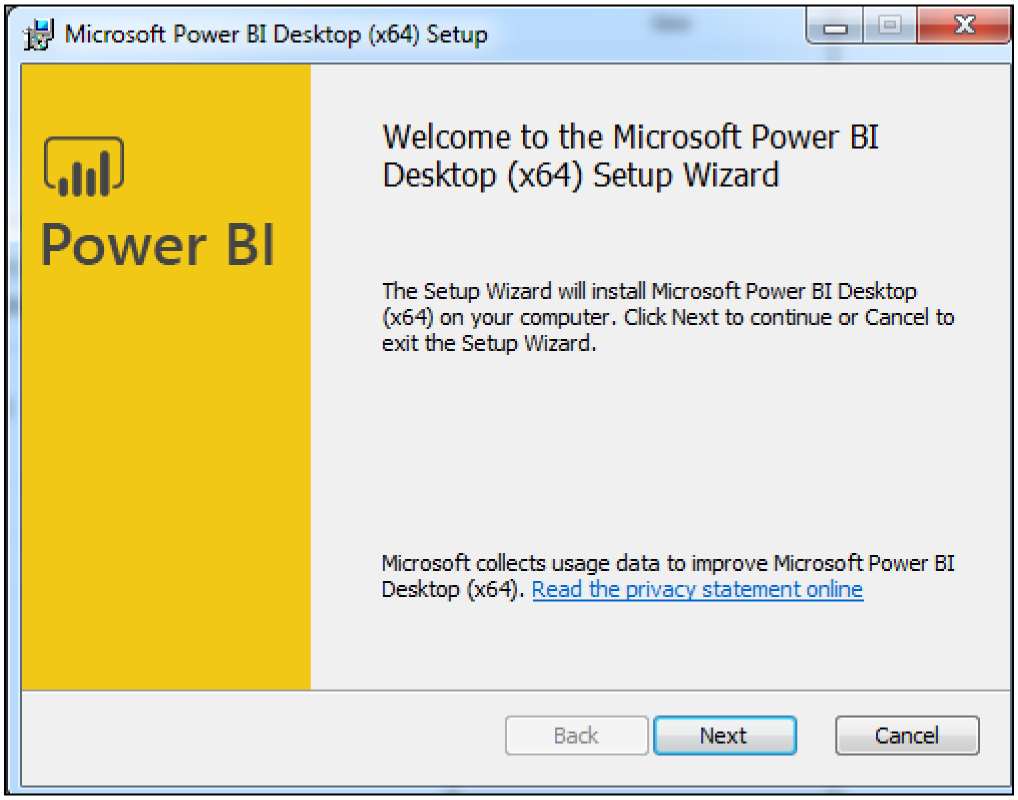
File Name	Size
PBIDesktop.msi	136.6 MB
✓ PBIDesktop_x64.msi	155.4 MB

Download Summary:

1. PBIDesktop_x64.msi

Total Size: 155.4 MB

When you run the installation file, following screen is displayed.

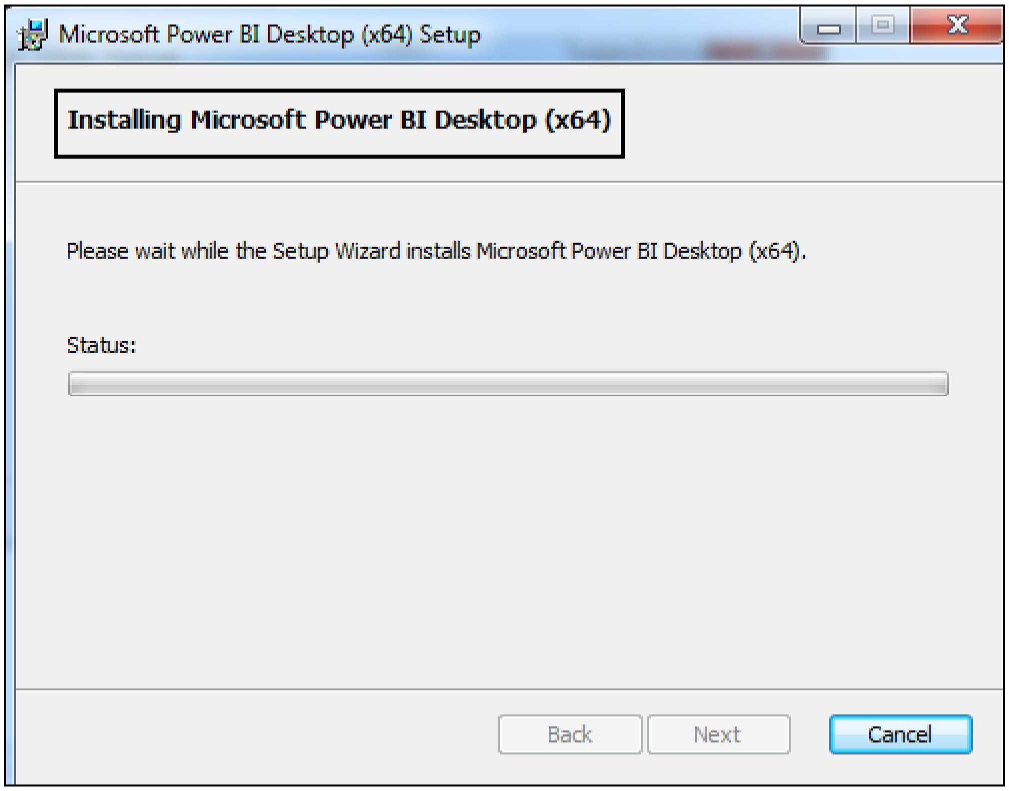


Welcome to the Microsoft Power BI Desktop (x64) Setup Wizard

The Setup Wizard will install Microsoft Power BI Desktop (x64) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Microsoft collects usage data to improve Microsoft Power BI Desktop (x64). Read the privacy statement online

Accept the license agreement and follow the instructions on the screen to finish the installation.

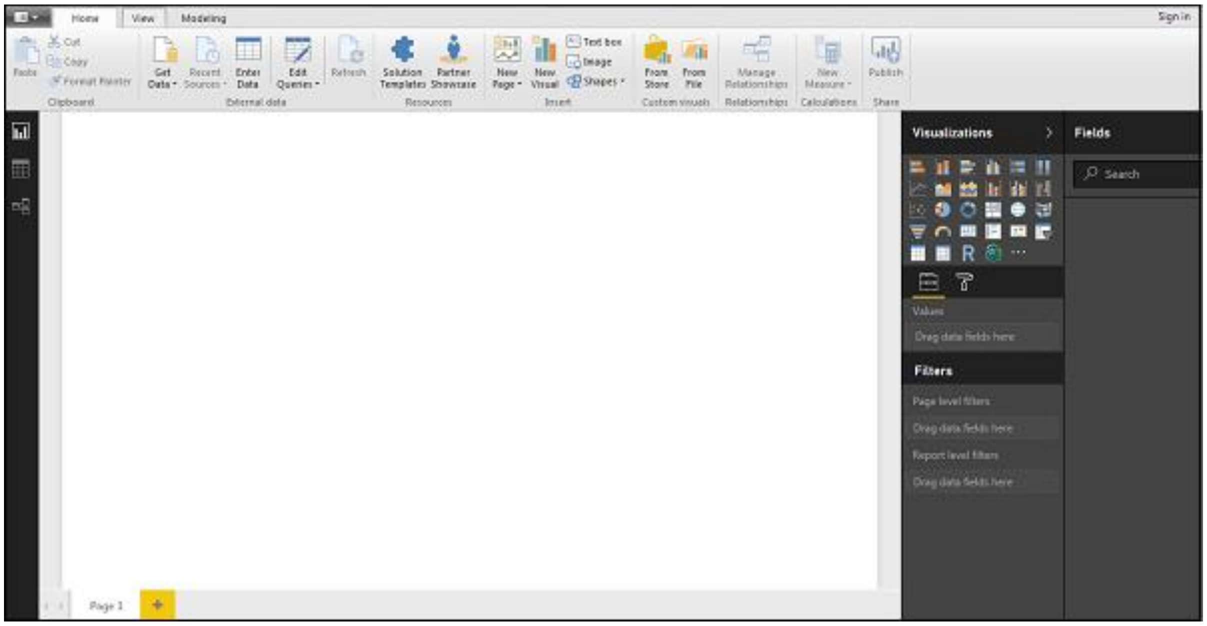


Installing Microsoft Power BI Desktop (x64)

Please wait while the Setup Wizard installs Microsoft Power BI Desktop (x64).

Status:

When Power BI is installed, it launches a welcome screen. This screen is used to launch different options related to get data, enrich the existing data models, create reports as well as publish and share reports.



Experiment 10 .

Objective

Studying Structures, different ways to declare define and initialize structures.

Theory

Power BI includes the following components −

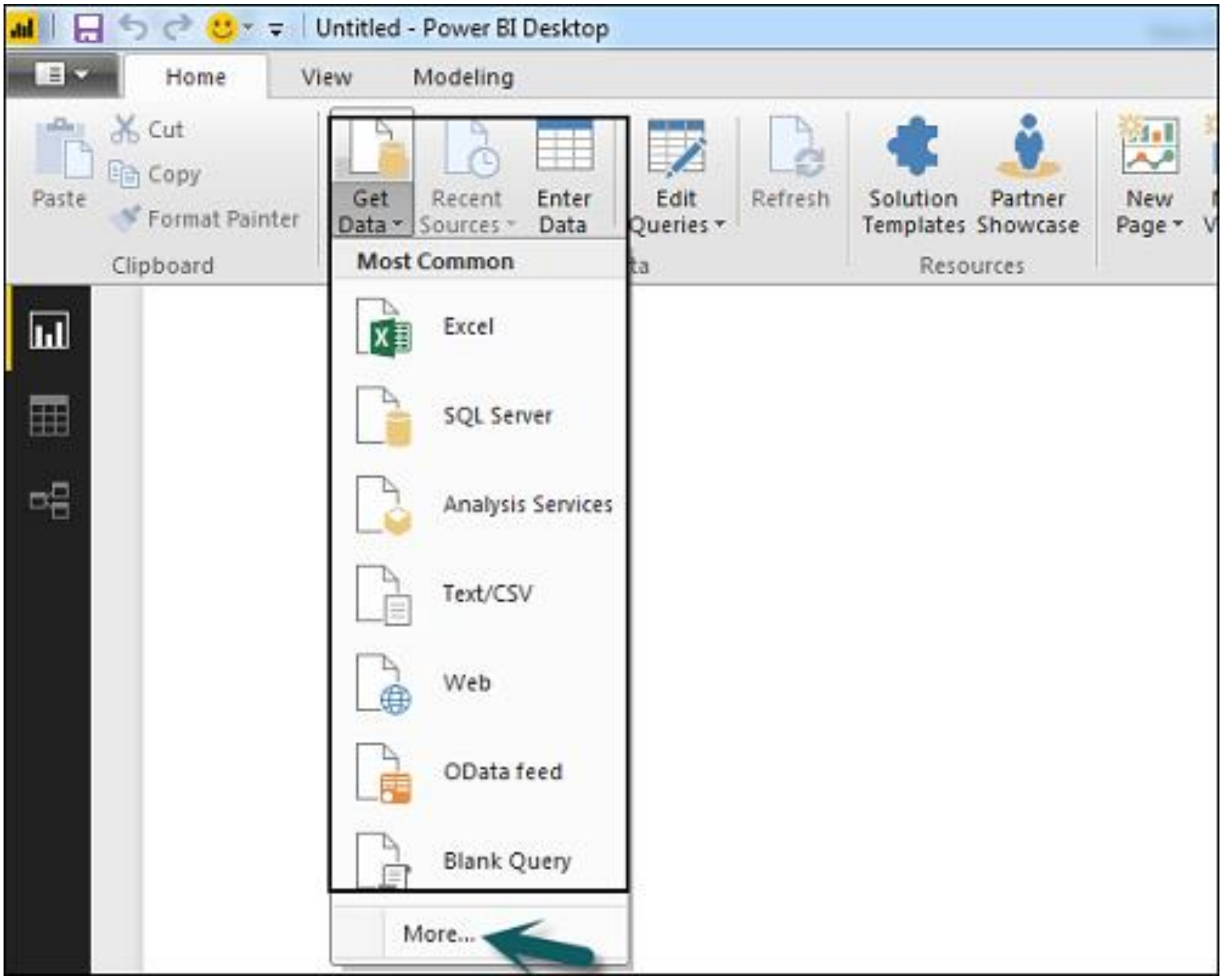
- **Power BI Desktop** − This is used to create reports and data visualizations on the dataset.
- **Power BI Gateway** − You can use Power BI on-premises gateway to keep your data fresh by connecting to your on-premises data sources without the need to move the data. It allows you to query large datasets and benefit from the existing investments.
- **Power BI Mobile Apps** − Using Power BI mobile apps, you can stay connected to their data from anywhere. Power BI apps are available for Windows, iOS, and Android platform.
- **Power BI Service** − This is a cloud service and is used to publish Power BI reports and data visualizations.

Power BI supports large range of data sources. You can click Get data and it shows you all the available data connections. It allows you to connect to different flat files, SQL database, and Azure cloud or even web platforms such as Facebook, Google Analytics, and Salesforce objects. It also includes ODBC connection to connect to other ODBC data sources, which are not listed.

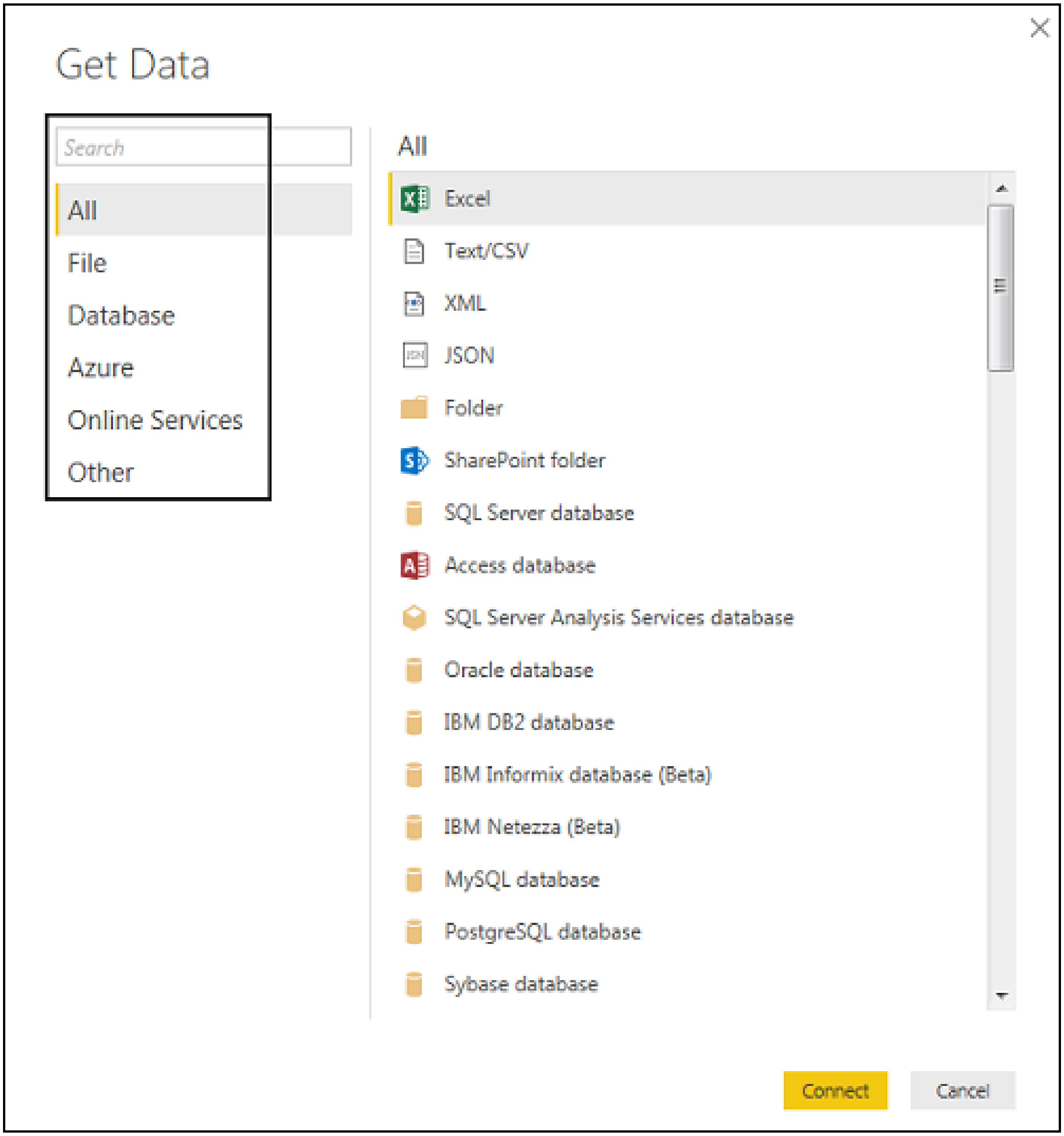
Following are the available data sources in Power BI −

- Flat Files
- SQL Database
- OData Feed
- Blank Query
- Azure Cloud platform
- Online Services
- Blank Query
- Other data sources such as Hadoop, Exchange, or Active Directory

To get data in Power BI desktop, you need to click the Get data option in the main screen. It shows you the most common data sources first. Then, click the More option to see a full list of available data sources.



When you click "More.." tab as shown in the above screenshot, you can see a new navigation window, where on the left side it shows a category of all available data sources. You also have an option to perform a search at the top.



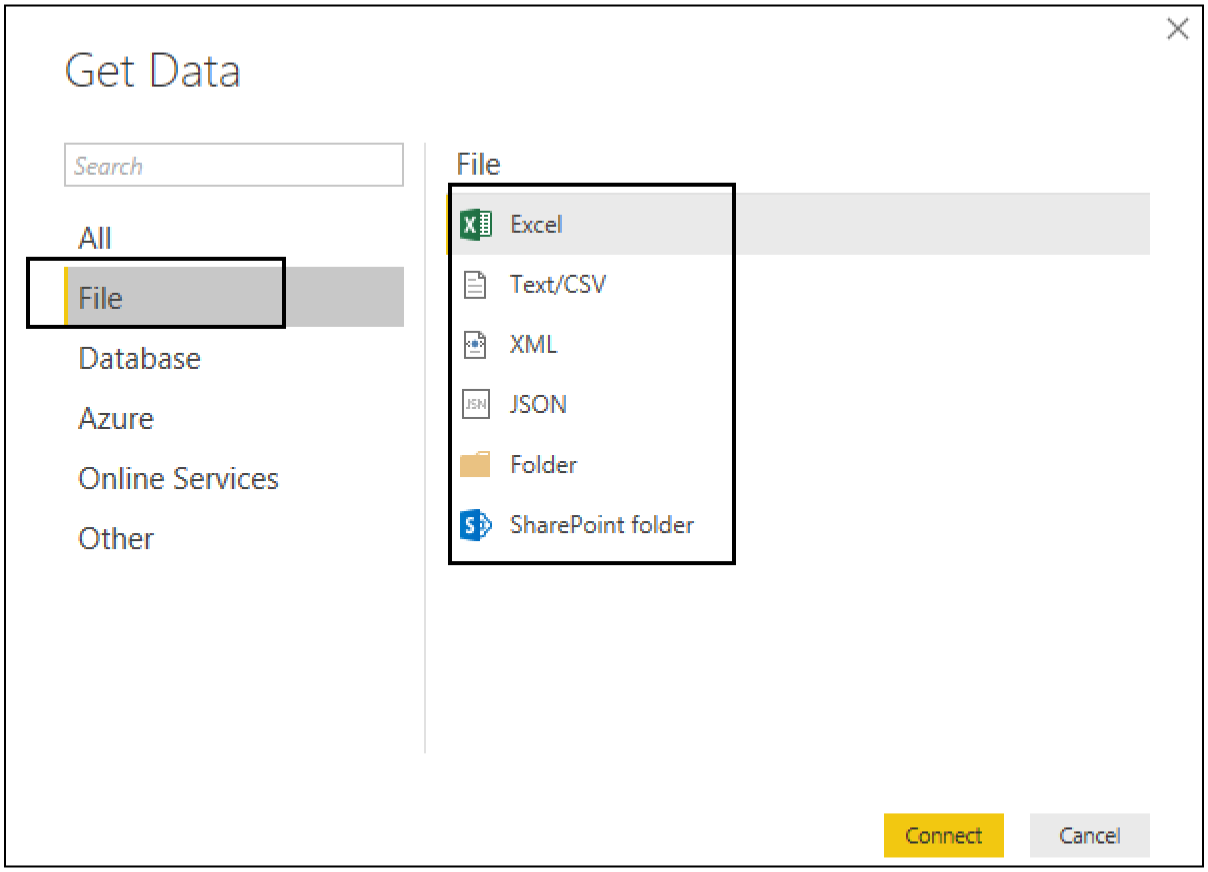
Following are the various **data sources** listed –

All

Under this category, you can see all the available data sources under Power BI desktop.

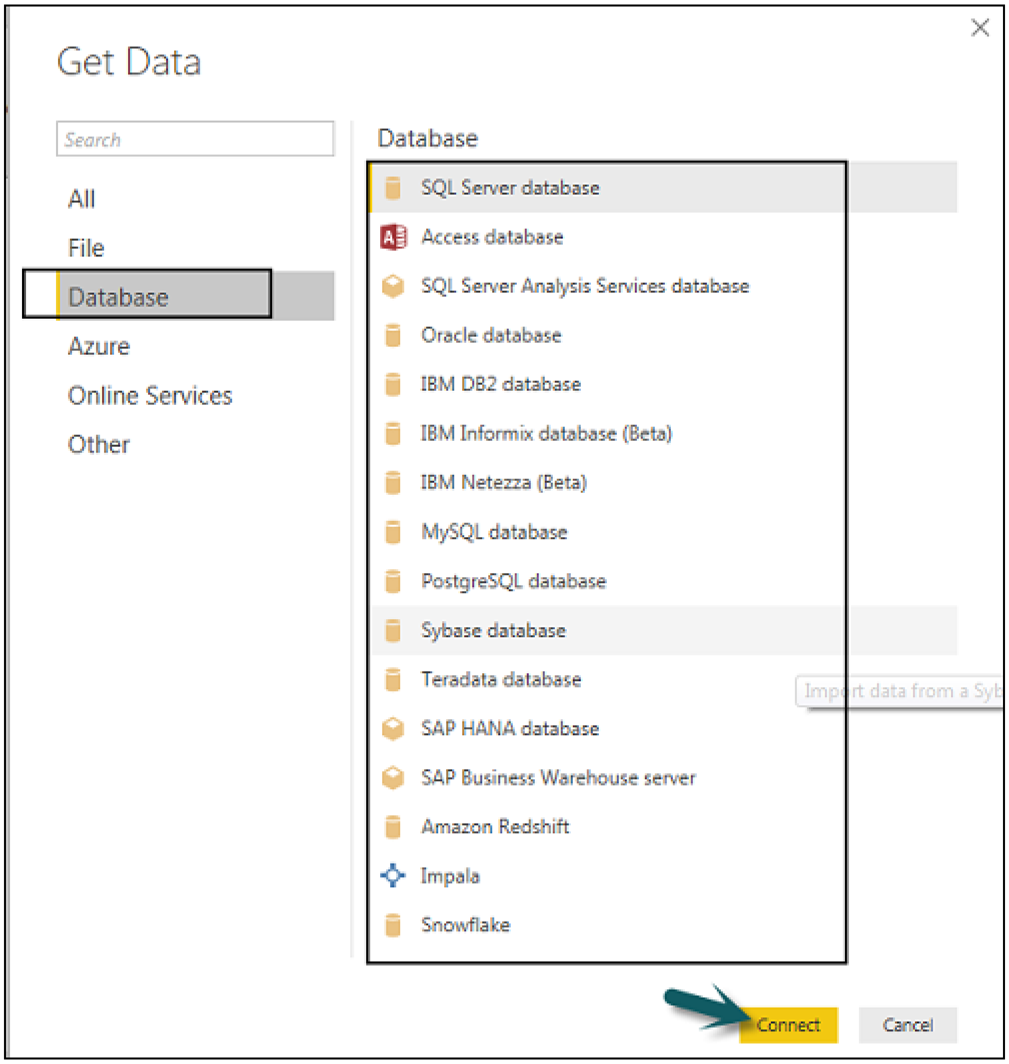
File

When you click File, it shows you all flat file types supported in Power BI desktop. To connect to any file type, select the file type from the list and click Connect. You have to provide the location of the file.



Database

When you click the Database option, it shows a list of all the database connections that you can connect to.



To connect to any database, select a Database type from the list as shown in the above screenshot. Click Connect.

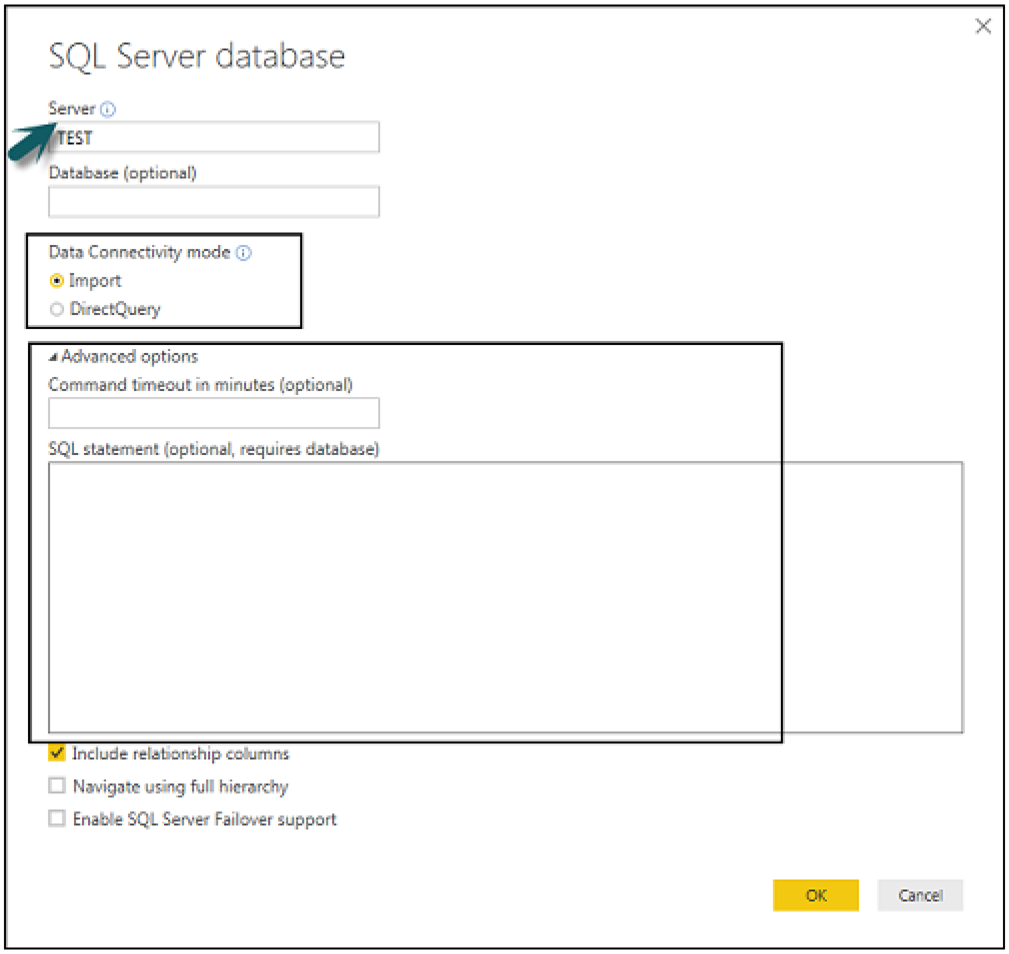
You have to pass Server name/ User name and password to connect. You can also connect via a direct SQL query using Advance options. You can also select Connectivity mode- Import or DirectQuery.

Note − You can't combine import and DirectQuery mode in a single report.

Import vs DirectQuery

DirectQuery option limits the option of data manipulation and the data stays in SQL database. DirectQuery is live and there is no need to schedule refresh as in the Import method.

Import method allows to perform data transformation and manipulation. When you publish the data to PBI service, limit is 1GB. It consumes and pushes data into Power BI Azure backend and data can be refreshed up to 8 times a day and a schedule can be set up for data refresh.


SQL Server database

Server: TEST

Database (optional)

Data Connectivity mode: [✓ Import] DirectQuery

Advanced options

Command timeout in minutes (optional)

SQL statement (optional, requires database)

[✓] Include relationship columns

[ ] Navigate using full hierarchy

[ ] Enable SQL Server Failover support

Advantages of Using DirectQuery

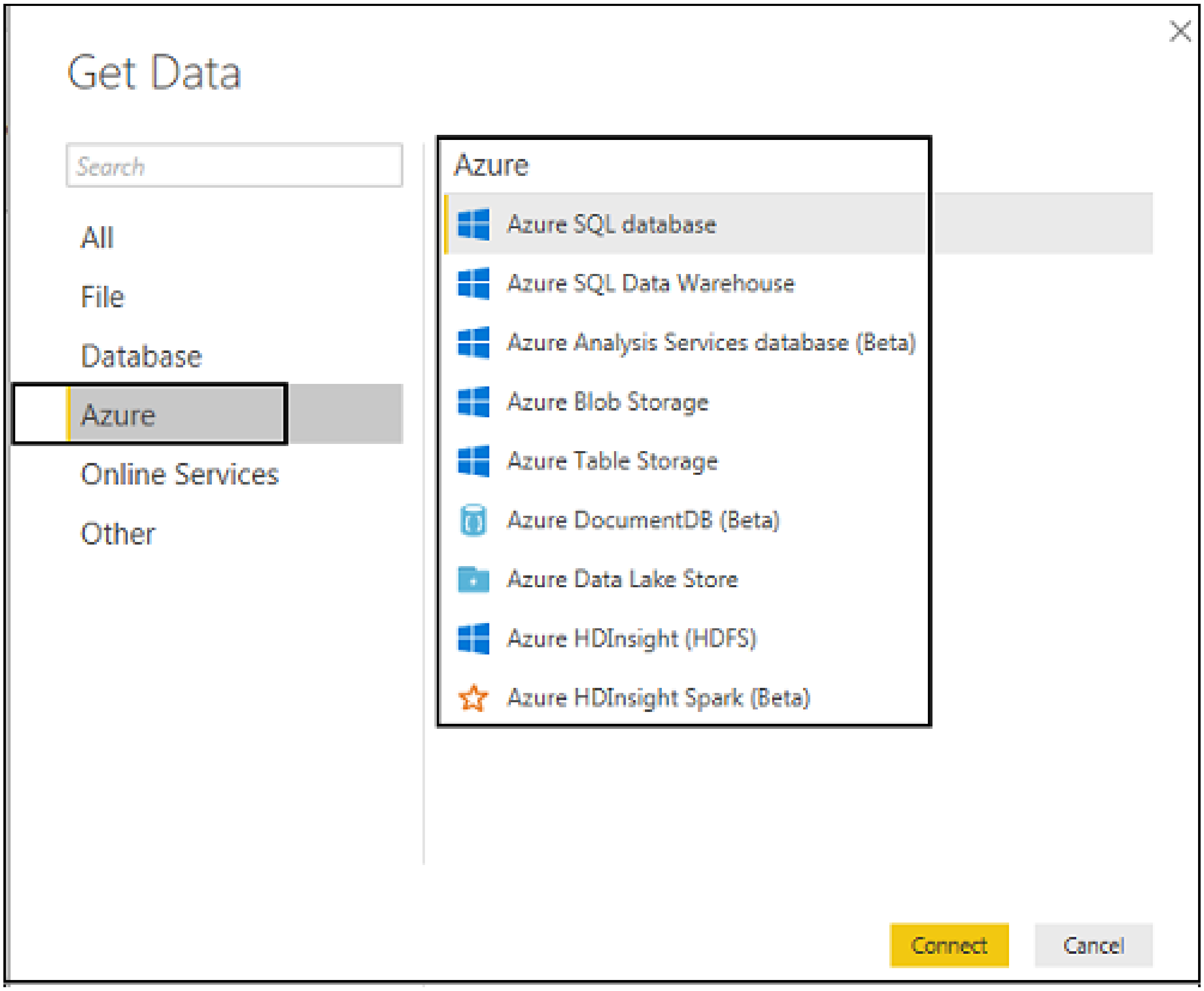
- Using DirectQuery, you can build data visualizations on large datasets, which is not feasible to import in Power BI desktop.
- DirectQuery doesn't apply any 1GB data set limit.
- With the use of DirectQuery, the report always shows current data.

Limitations of Using DirectQuery

- There is a limitation of 1 million row for returning data while using DirectQuery. You can perform aggregation of more number of rows, however, the result rows should be less than 1 million to return the dataset.
- In DirectQuery, all tables should come from a single database.
- When a complex query is used in the Query editor, it throws an error. To run a query, you need to remove the error from the query.
- In DirectQuery, you can use Relationship filtering only in one direction.
- It doesn't support special treatment for time-related data in tables.

Azure

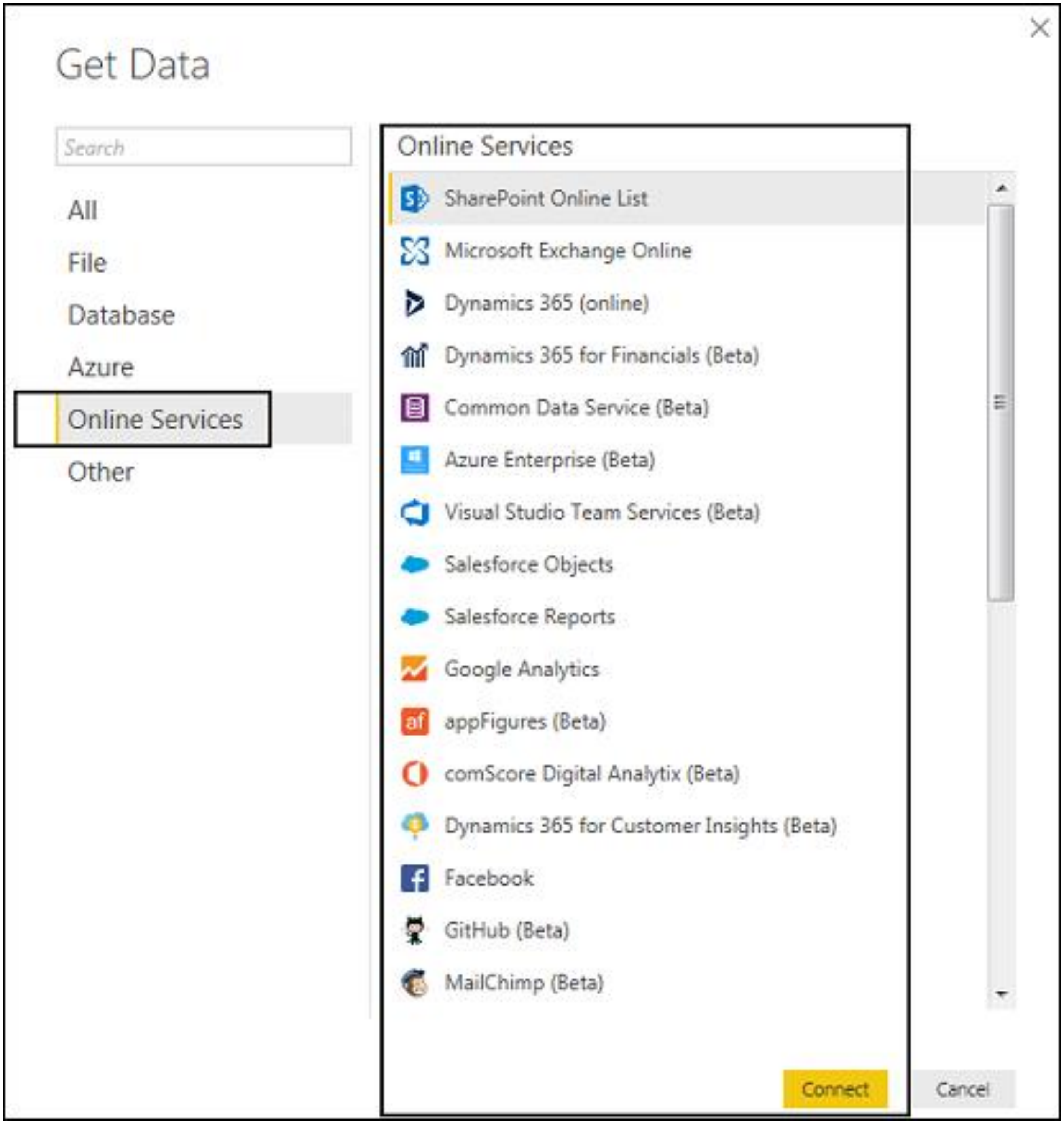
Using the Azure option, you can connect to the database in Azure cloud. Following screenshot shows the various options available under Azure category.

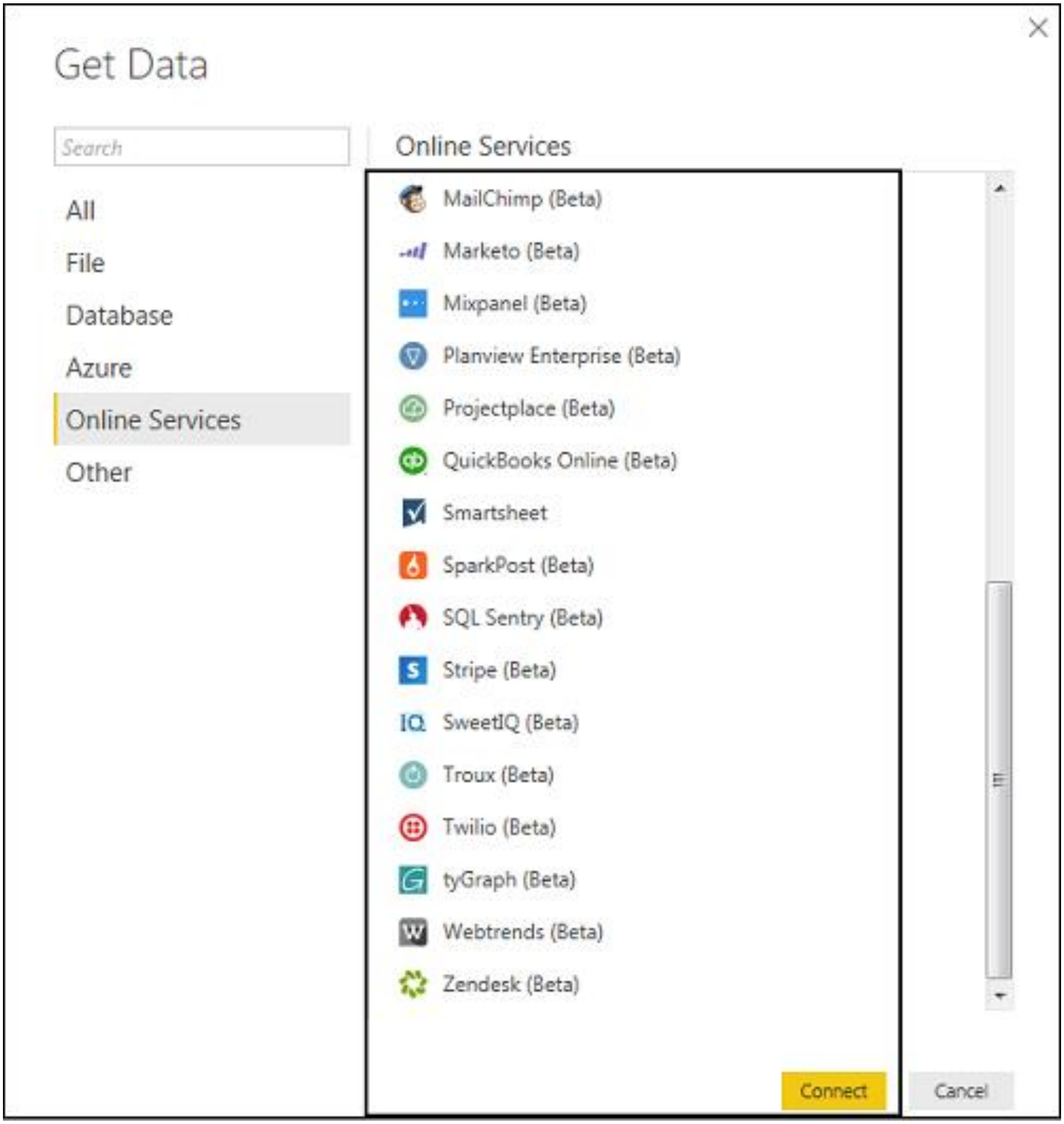


Online Services

Power BI also allows you to connect to different online services such as Exchange, Salesforce, Google Analytics, and Facebook.

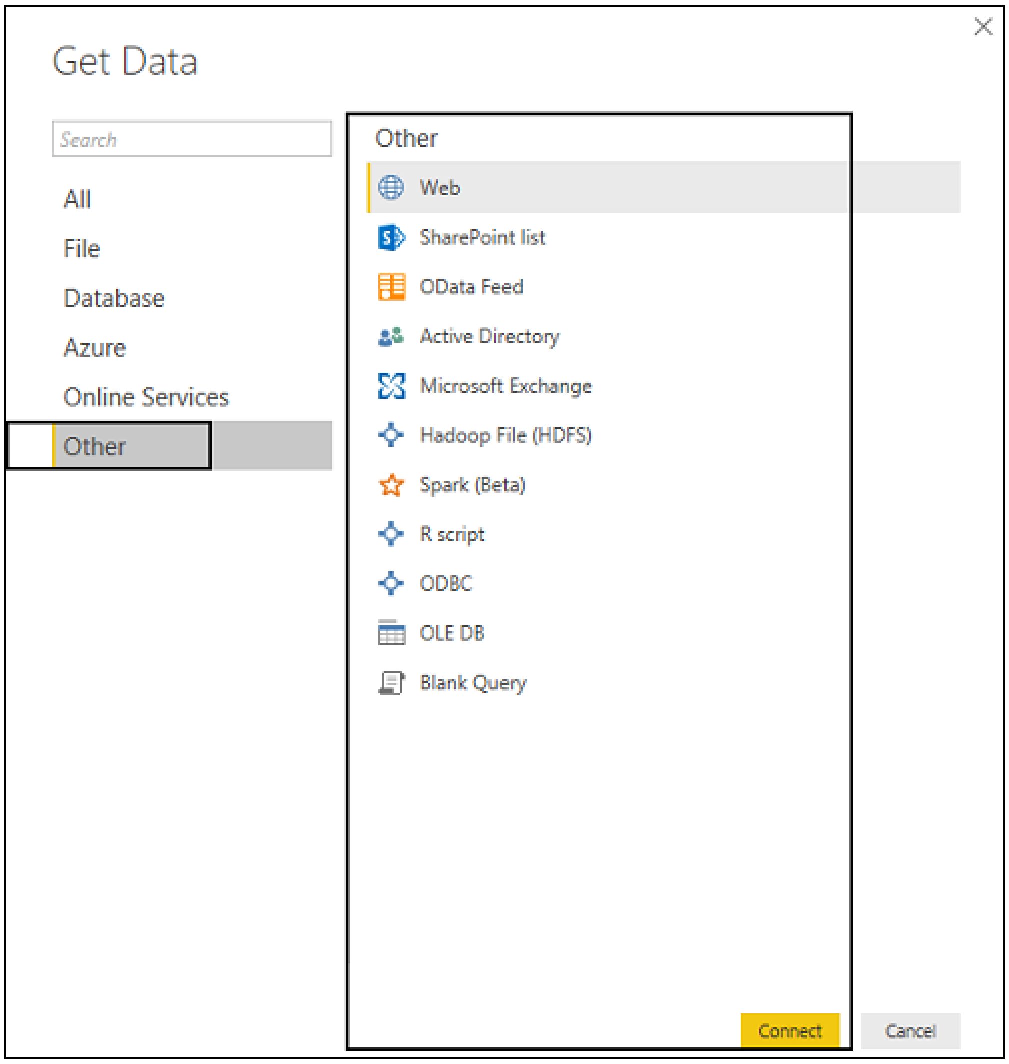
Following screenshots shown the various options available under Online Services.




Get Data
Search
All
File
Database
Azure
Online Services
Other
Online Services
MailChimp (Beta)
Marketo (Beta)
Mixpanel (Beta)
Planview Enterprise (Beta)
Projectplace (Beta)
QuickBooks Online (Beta)
Smartsheet
SparkPost (Beta)
SQL Sentry (Beta)
Stripe (Beta)
SweetIQ (Beta)
Troux (Beta)
Twilio (Beta)
tyGraph (Beta)
Webtrends (Beta)
Zendesk (Beta)
Connect
Cancel

Other

Following screenshot shows the various options available under other category.



Experiment 11 .

Objective

Transforming Data & Interactive Data Exploration.

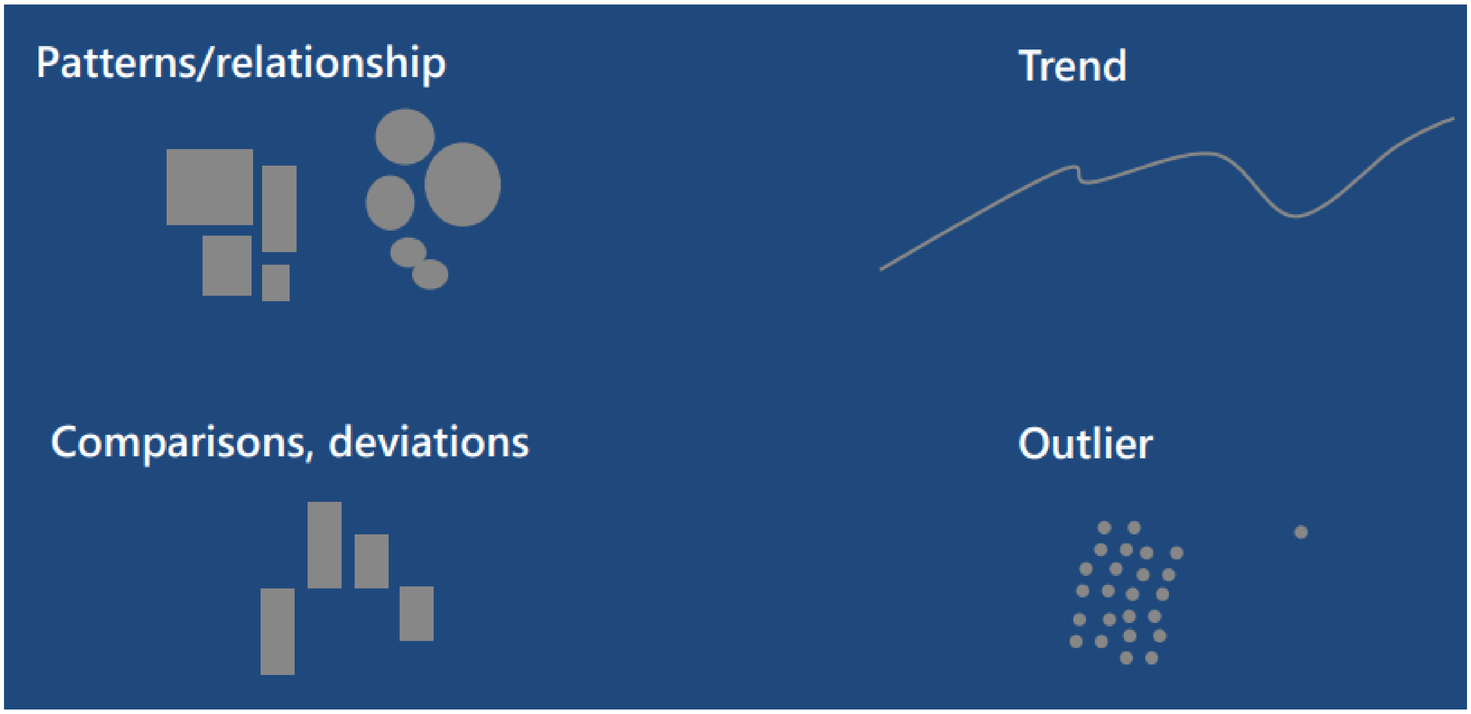
Theory

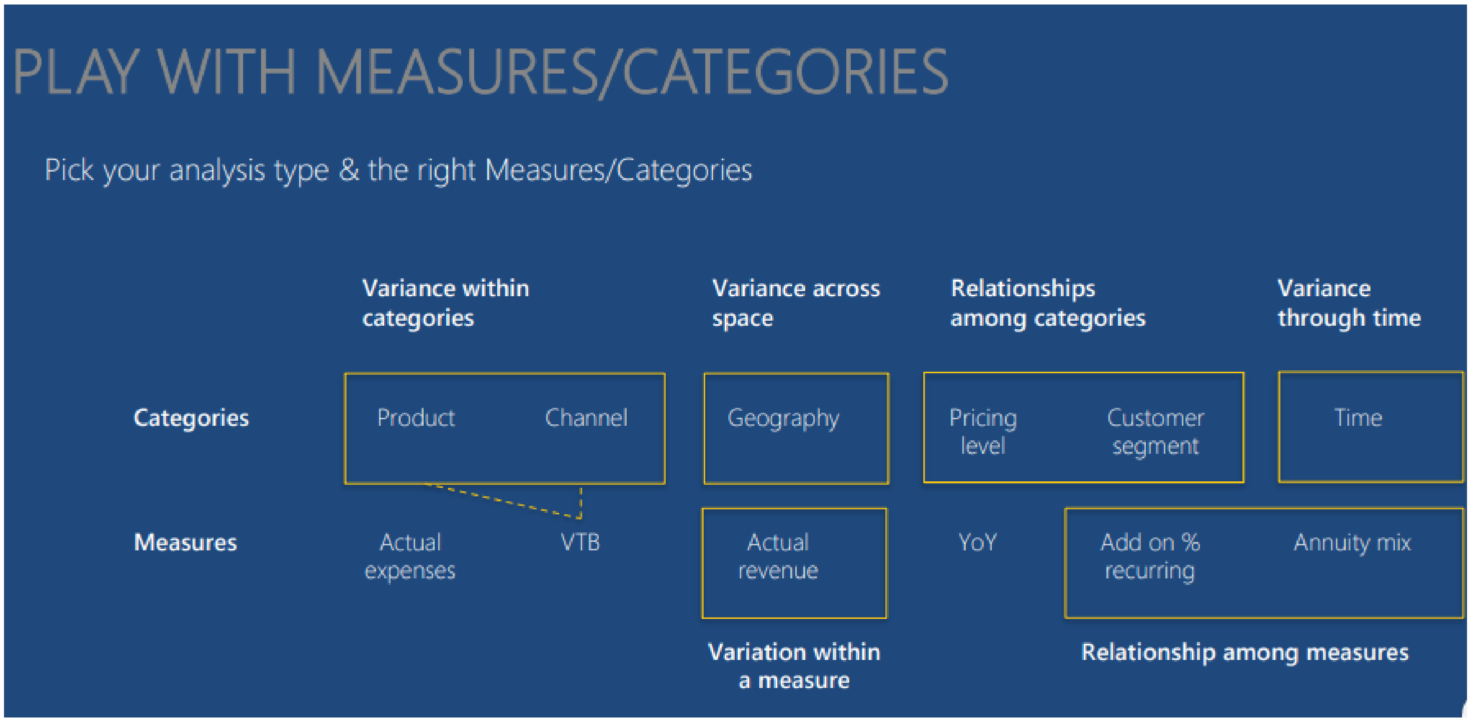
The process of creating insight from data by understanding the topic
• From a strategic perspective
• In the context which the analysis occurs
• Key objectives and desired outcomes
• Audience - their perspective and needs

Here, **type** is the pointer's base type; it must be a valid C data type and var-name is the name of the pointervariable. The asterisk * used to declare a pointer. In this statement the asterisk is being used to designate a variable as a pointer.

Exercise:

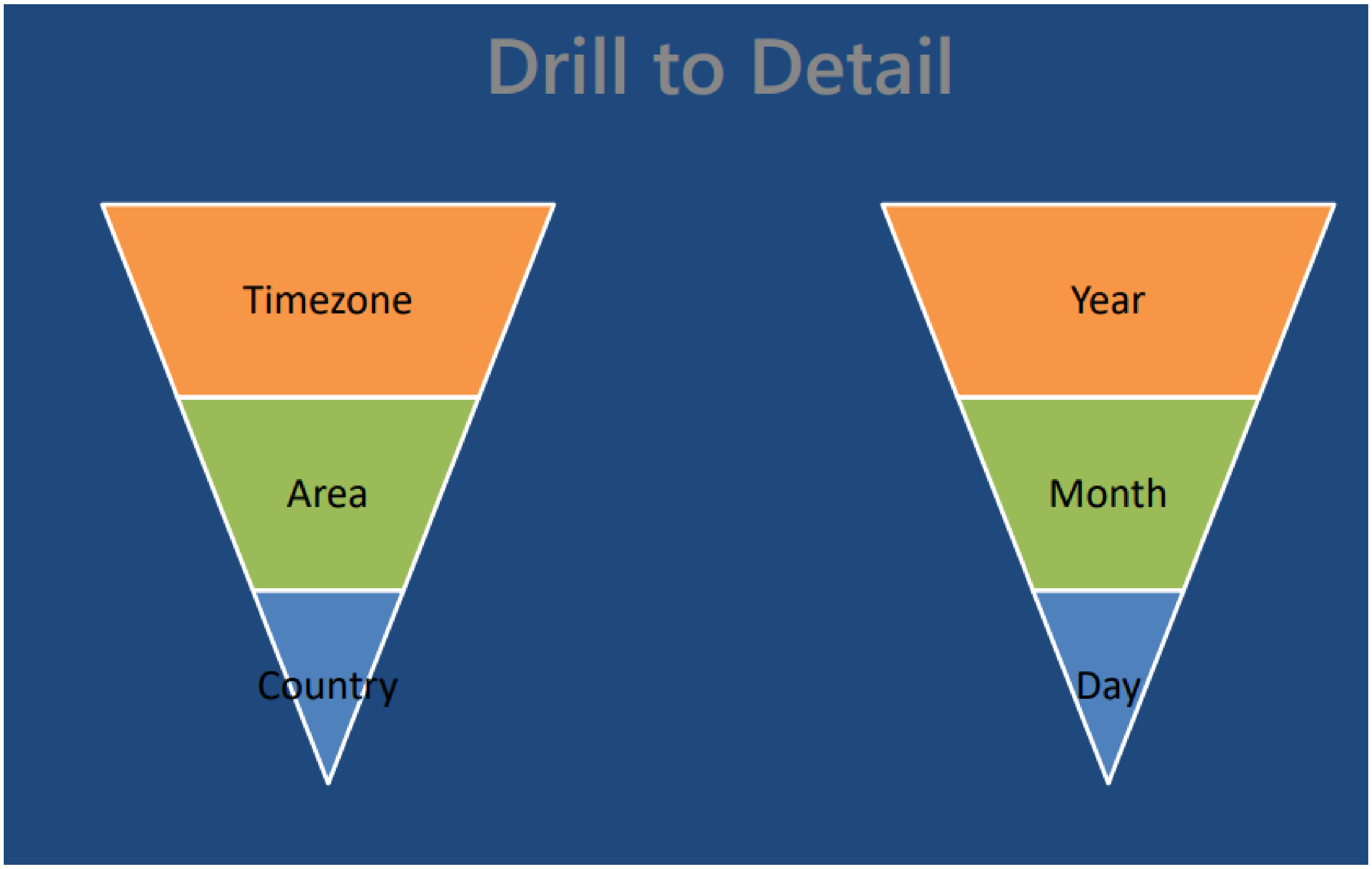
Familiarize yourself with the different elements of the data
• Dimensions
• Geography, Time, Product, Customer, Industry
• Measures and KPIs • Actual, Budget, VTB, YoY, % Attained
• Hierarchies • Year – Half – Quarter – Month – Week – Day





PLAY WITH MEASURES/CATEGORIES

Pick your analysis type & the right Measures/Categories



Drill to Detail

Experiment 12

Objective

Cyclomatic complexity for finding source code complexity.

Theory

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

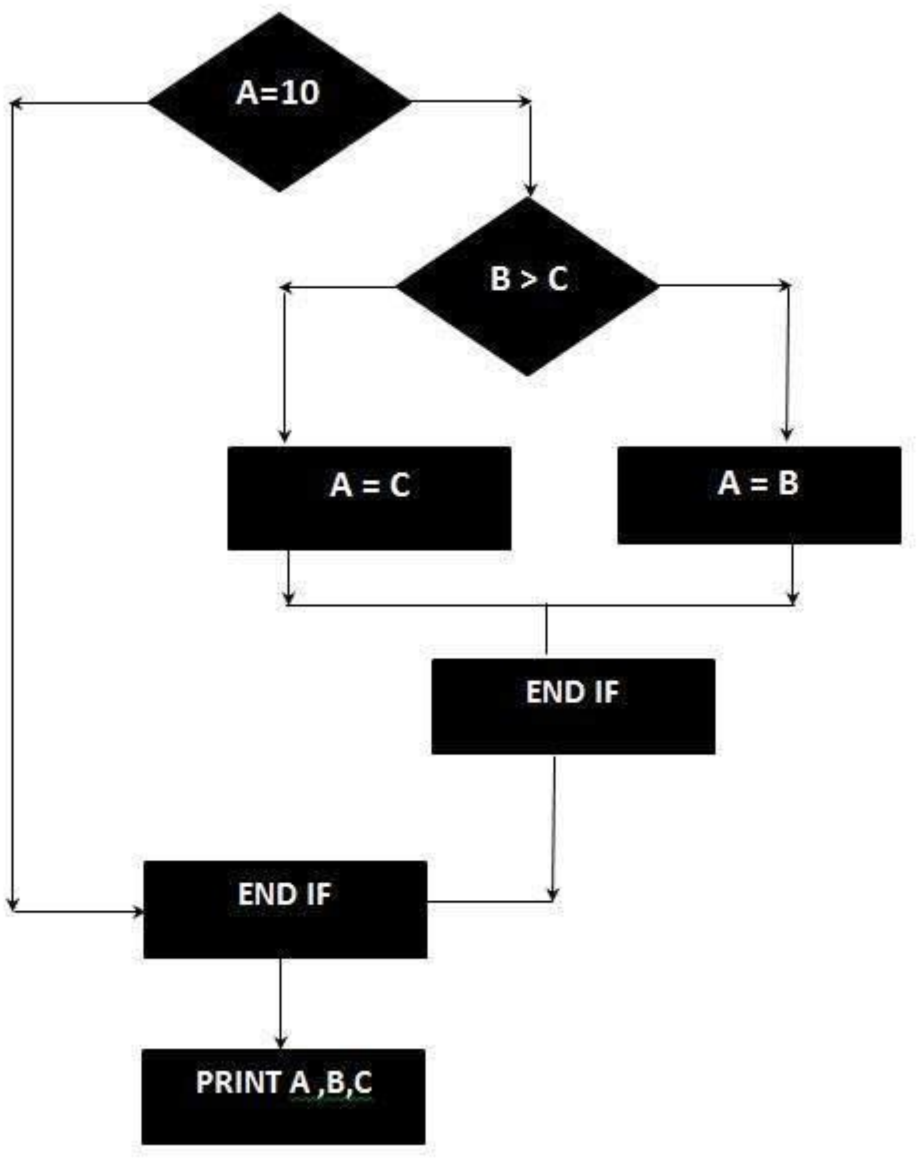
Lower the Program's cyclomatic complexity, lower the risk to modify and easier to understand. It can be represented using the below formula:

```
Cyclomatic complexity = E - N + 2*P

where,

  E = number of edges in the flow graph.

  N = number of nodes in the flow graph.

  P = number of nodes that have exit points
```

Example :

```
IF A = 10 THEN

 IF B > C THEN

   A = B

 ELSE

   A = C

 ENDIF

ENDIF

Print A

Print B

Print C
```

FlowGraph:



The Cyclomatic complexity is calculated using the above control flow diagram that shows seven nodes(shapes) and eight edges (lines), hence the cyclomatic complexity is 8 - 7 + 2 = 3

```
import sys
import re
import os
from subprocess import Popen, PIPE

def complexity_for_file(filename):
    cmd = '%s %s' % (PATH_TO_COMPLEXITY, filename)
    p = Popen(cmd, shell=True, stdout=PIPE, stderr=PIPE, close_fds=True)
    errors = p.stderr.read()
    if errors:
        sys.exit(errors)
    tuples = []
    reached_functions = False
    function_header = re.compile('Funcname|Classless_Function')
    for line in p.stdout.readlines():
        if function_header.match(line):
            reached_functions = True
        if reached_functions:
            fields = line.split()
            # We don't want the header lines.
            try:
                int(fields[5])
            except:
                continue
            # filename, function name, lines of code, and complexity.
            tuples.append((filename, fields[0], fields[4], fields[5]))
    return tuples

def add_to_functions(functions, dir, filenames):
    for filename in filenames:
        if filename.endswith('.py'):
            functions += complexity_for_file(os.path.join(dir, filename))

def get_column_widths(function_tuples):
    widths = []
    if not function_tuples:
        return widths
    for col in range(len(function_tuples[0])):
        widths.append(max([len(ft[col]) for ft in function_tuples]))
    return widths

def get_print_parms(widths, function_tuple):
    parms = []
    for x in range(len(widths)):
        parms.append(widths[x])
        parms.append(function_tuple[x])
    return tuple(parms)

def main():
    function_tuples = []
    paths = sys.argv[1:] or os.path.curdir
    for path in paths:
        path = os.path.expanduser(path)
        if os.path.isdir(path):
            os.path.walk(path, add_to_functions, function_tuples)
        elif os.path.isfile(path):
            function_tuples += complexity_for_file(path)
    # Filter out functions less than desired complexity level.
    function_tuples = [t for t in function_tuples if int(t[-1]) &gt;=
COMPLEXITY_LEVEL]
    # Sort by complexity.
    function_tuples.sort(key=lambda t: int(t[-1]))
    function_tuples.reverse()
    print "\nShowing functions/methods with complexity greater than or equal to
%s:\n" % COMPLEXITY_LEVEL
    if function_tuples:
        headers = ('Filename', 'Function/Method', 'Lines of Code',
'Complexity')
        widths = get_column_widths(function_tuples + [headers])
        print '%-*s %-*s %*s %*s' % get_print_parms(widths, headers)
        for ft in function_tuples:
            print '%-*s %-*s %*s %*s' % get_print_parms(widths, ft)
    else:
        print "None."

if __name__ == "__main__":
    sys.exit(main())
```

Experiment 13

Objective

Creating a tool for finding code complexity, sloc, bloc , number of operators.

Theory

Abstract:

In this project we analyze different project after the analyzation of project we represent the complexity of project by graphically way. The project is created on .Net Platform. On run time we select a project on directory after identify the path of the project we get the complexity of the project.

Project Description:

In this project initially we select C# to calculate Cyclomatic Complexity and Halstead's Complexity automatically by reading the ".cs" file from RichTextBox all the code will be analyzed and the response will be generated on labels, it can also be plot on charts. On run time we can analyze single ".cs" file.

Cyclomatic Complexity:

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

Lower the Program's cyclomatic complexity, lower the risk to modify and easier to understand. It can be represented using the below formula:

Cyclomatic complexity = E - N + 2*P
where,
E = number of edges in the flow graph.
N = number of nodes in the flow graph.
P = number of nodes that have exit points

Halstead's Complexity Measures:

According to Halstead, "A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands". Halstead metrics think a program as sequence of operators and their associated operands.
It defines various indicators to check complexity of module.

Parameter	Meaning
n1	Number of unique operators
n2	Number of unique operands
N1	Number of total occurrence of operators
N2	Number of total occurrence of operands

Metric	Meaning	Mathematical Representation
n	Vocabulary	n1 + n2
N	Size	N1 + N2
V	Volume	Length * Log2 Vocabulary
D	Difficulty	(n1/2) * (N1/n2)
E	Efforts	Difficulty * Volume
B	Errors	Volume / 3000

Namespace name:

CSharpMcCabeHalstead

Project Classes:

1. Halstead.cs
2. McCabe.cs
3. SourceCodeEditor.cs
4. Program.cs
5. CSharpMcCabeHalstead.cs

Features (Tool Box Components):

They are listed below.

1. Charts
2. Rich Text Box
3. Windows Form
4. Buttons
5. Tool tip
6. Labels

Libraries Used:

- using System;
- using System.IO;
- using System.Windows.Forms;
- using System.Text.RegularExpressions;System.Threading.Tasks
- System.Collection.Genric
- System.ComponentModel
- System.Text

Experiment 14

Objective

Creating a tool for finding code complexity, sloc, bloc , number of operators continue

<u>Classes:</u>

Halstead.cs

```
using System;
using System.Text.RegularExpressions;

namespace CSharpMcCabeHalstead
{
    class Halstead
    {
        private string _sourceCode;
        private string _operators;
        private string _notOperators;
        private int _totalNumberOfOperators;
        private int _totalNumberOfOperands;
        private int _numberOfDistinctOperators;
        private int _numberOfDistinctOperands;

        public Halstead(string sourceCode)
        {
            this._sourceCode = sourceCode;
        }

        public int CountNumberOfDistinctOperators()
        {
            var operatorsPattern = this._operators;
            var operatorsMatches = Regex.Matches(this._sourceCode, operatorsPattern);

            this._numberOfDistinctOperators = 0;
            for (var currentMatchCount = 0; currentMatchCount < operatorsMatches.Count;
currentMatchCount++)
            {
                var countRepetitiveOperators = 0;
                for (var nextMatchCount = currentMatchCount + 1; nextMatchCount < operatorsMatches.Count;
nextMatchCount++)
                {
                    if (operatorsMatches[currentMatchCount].Value !=
operatorsMatches[nextMatchCount].Value)
                    {
                        continue;
                    }

                    countRepetitiveOperators++;
                }

                if (countRepetitiveOperators == 0)
                {
                    this._numberOfDistinctOperators++;
                }
            }

            return this._numberOfDistinctOperators;
        }

        public int CountNumberOfDistinctOperands()
        {
            const string replacementCode = " ";

            var operandsPattern = this._notOperators;
            var operandsRegex = new Regex(operandsPattern);
            this._sourceCode = operandsRegex.Replace(this._sourceCode, replacementCode);

            operandsPattern = @"(\b\w+\b)";
            var operandsMatches = Regex.Matches(this._sourceCode, operandsPattern);

            this._numberOfDistinctOperands = 0;
            for (var currentMatchCount = 0; currentMatchCount < operandsMatches.Count;
currentMatchCount++)
            {
                var countReapeatOperands = 0;
                for (var nextMatchCount = currentMatchCount + 1; nextMatchCount < operandsMatches.Count;
nextMatchCount++)
                {
                    if (operandsMatches[currentMatchCount].Value !=
operandsMatches[nextMatchCount].Value)
                    {
                        continue;
                    }

                    countReapeatOperands++;
                }

                if (countReapeatOperands == 0)
                {
                    this._numberOfDistinctOperands++;
                }
            }

            return this._numberOfDistinctOperands;
        }

        public int CountTotalNumberOfOperators()
        {
            var operatorsPattern = @"(\b.*\s\?\s.*\s\:\s.*)";
            var operatorsMatches = Regex.Matches(this._sourceCode, operatorsPattern);
            this._totalNumberOfOperators = operatorsMatches.Count;
            this._operators = operatorsPattern + "|";

            operatorsPattern =
@"(~)|(\(int\))|(\(float\))|(\(string\))|(\(array\))|(\(object\))|(\(bool\))";
            operatorsMatches = Regex.Matches(this._sourceCode, operatorsPattern);
            this._totalNumberOfOperators += operatorsMatches.Count;
            this._operators += operatorsPattern + "|";

            operatorsPattern = @"(\+{1,2})|(\-
{1,2})|(\*{1,2})|(\/)|(%)|(\={1,3})|(!=)|(>{1,2})|(<{1,2})|(>=)|(<=)|(<=>)|(\+=)|(\-
=)|(\*=)|(\/=)|(%=)|(\*{2}=)|(&{1,2})|(\&{2}=)|(\|{1,2})|(\|{2}=)|(\^)|(~)|(=~)|(\.{1,3})|([\s\S]\,[\s])|
(!)|(!~)|({)";
            operatorsMatches = Regex.Matches(this._sourceCode, operatorsPattern);
            this._totalNumberOfOperators += operatorsMatches.Count;
            this._operators += operatorsPattern + "|";

            operatorsPattern =
@"(\bdelete\b)|(\bin\b)|(\binctanceof\b)|(\bnew\b)|(\bthis\b)|(\btypeof\b)|(\bvoid\b)|(\bgoto\b)";
            operatorsMatches = Regex.Matches(this._sourceCode, operatorsPattern);
            this._totalNumberOfOperators += operatorsMatches.Count;
            this._operators += operatorsPattern;

            return this._totalNumberOfOperators;
        }

        public int CountTotalNumberOfOperands()
        {
            const string replacementCode = " ";

            this._notOperators = this._operators;
            this._totalNumberOfOperands = 0;

            var operandsPattern =
@"(\babstract\b)|(\bbreak\b)|(\bchar\b)|(\bcontinue\b)|(\bdo\b)|(\bevent\b)|(\bfinally\b)|(\bforeach\b)|(
\bIn\b)|(\binternal\b)|(\bnamespace\b)|(\boperator\b)|(\bparams\b)|(\breadonly\b)|(\bsealed\b)|(\bstatic\
b)|(\bthis\b)|(\btypeof\b)|(\bunsafe\b)|(\bvoid\b)|(\bas\b)|(\bbyte\b)|(\bchecked\b)|(\bdecimal\b)|(\bdou
ble\b)|(\bexplicit\b)|(\bfixed\b)|(\bgoto\b)|(\bin\b)|(\bis\b)|(\bnew\b)|(\bout\b)|(\bprivate\b)|(\bref\b
)|(\bshort\b)|(\bstring\b)|(\bstring\b)|(\bstring\b)|(\bstring\b)|(\bstring\b)|(\bstring\b)|(\bstring\b)|
(\bstring\b)|(\bstring\b)|(\bthrow\b)|(\buint\b)|(\bushort\b)|(\bvolatile\b)|(\bbase\b)|(\bcase\b)|(\bcla
ss\b)|(\bfloat\b)|(\bif\b)|(\bint\b)|(\block\b)|(\bNull\b)|(\bprotected\b)|(\breturn\b)|(\bsizeof\b)|(\bs
truct\b)|(\btrue\b)|(\bbulong\b)|(\busing\b)|(\bwhile\b)|(\bbool\b)|(\bcatch\b)|(\bconst\b)|(\bdelegate\b
)|(\benum\b)|(\bfalse\b)|(\bfor\b)|(\bimplicit\b)|(\binterface\b)|(\blong\b)|(\bObject\b)|(\boverride\b)|
(\bpublic\b)|(\bsbyte\b)|(\bstackalloc\b)|(\bswitch\b)|(\btry\b)|(\bunchecked\b)|(\bvirtual\b)|(\badd\b)|
(\basync\b)|(\bdynamic\b)|(\bglobal\b)|(\bjoin\b)|(\bpartial\b)|(\bselect\b)|(\bvar\b)|(\byield\b)|(\bali
as\b)|(\bawait\b)|(\bFROM\b)|(\bgroup\b)|(\blet\b)|(\bset\b)|(\bwhere\b)|(\bascending\b)|(\bdescending\b)
|(\bdescending\b)|(\binto\b)|(\borderby\b)|(\bremove\b)|(\bvalue\b)";
            this._notOperators += operandsPattern;

            operandsPattern = this._notOperators;
            var operandsRegex = new Regex(operandsPattern);
            this._sourceCode = operandsRegex.Replace(this._sourceCode, replacementCode);

            operandsPattern = @"(\b\w+\b)";
            var matches = Regex.Matches(this._sourceCode, operandsPattern);
            this._totalNumberOfOperands = matches.Count;

            return this._totalNumberOfOperands;
        }

        public int CalculateProgramVocabulary()
        {
            return (this._numberOfDistinctOperators + this._numberOfDistinctOperands);
        }

        public int CalculateProgramLength()
        {
            return (this._totalNumberOfOperators + this._totalNumberOfOperands);
        }


        public double CalculateProgramVolume()
        {
            return (this.CalculateProgramLength() * Math.Log(this.CalculateProgramVocabulary(), 2));
        }


        public double CalculateProgramDifficulty()
        {
            return (this._numberOfDistinctOperators / 2 * this._totalNumberOfOperands /
_numberOfDistinctOperands);
        }

        public double CalculateProgramEffort()
        {
            return (this.CalculateProgramVolume() * this.CalculateProgramDifficulty());
        }

        public double CalculateProgramErrors()
        {
            return (this.CalculateProgramVolume() / 3000);
        }

    }
```

```
}
```

McCabe.cs

```
using System.Text.RegularExpressions;

namespace CSharpMcCabeHalstead
{
    class McCabe
    {
        private readonly string _sourceCode;

        public McCabe(string sourceCode)
        {
            this._sourceCode = sourceCode;
        }

        public int CalculateCyclomaticComplexity()
        {
            const int numberOfConnectedComponentsCoefficient = 2;

            var numberOfGraphVertices = 0;
            var numberOfGraphArcs = 0;

            var loopsPattern = @"(\bswitch\b)";
            var loopsMatches = Regex.Matches(this._sourceCode, loopsPattern);
            numberOfGraphVertices += loopsMatches.Count;

            loopsPattern = @"(\bcase\b)|(\bdefault\b)|(\bfor\b)|(\bforeach\b)|(\bwhile\b)";
            loopsMatches = Regex.Matches(this._sourceCode, loopsPattern);
            numberOfGraphArcs += loopsMatches.Count;

            loopsPattern = @"(\bif\b)";
            loopsMatches = Regex.Matches(this._sourceCode, loopsPattern);
            numberOfGraphVertices += loopsMatches.Count;
            numberOfGraphArcs += 2 * loopsMatches.Count;

            numberOfGraphVertices++;
            return numberOfGraphArcs - numberOfGraphVertices + numberOfConnectedComponentsCoefficient;
        }
    }
}
```

Program.cs

```
using System;
using System.Windows.Forms;

namespace CSharpMcCabeHalstead
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new CSharpMcCabeHalstead());
        }
    }
}
```

SourceCodeEditor.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Text.RegularExpressions;

namespace CSharpMcCabeHalstead
{
    class SourceCodeEditor
    {
        private string _sourceCode;

        public string RemoveUnnecessaryCharacters(string sourceCode)
        {
            return RemoveComments(RemoveMultilineComments(sourceCode));
        }

        private static string RemoveComments(string sourceCode)
        {
            return Regex.Replace(sourceCode, @"(?is)\s\/\*.+?\\*\/\s", String.Empty);
        }

        private static string RemoveMultilineComments(string sourceCode)
        {
            return Regex.Replace(sourceCode, @"\s\/\/.+", String.Empty);
        }

        public void OpenFromFile(OpenFileDialog openFileDialog, TextBox sourceCodeTextBox)
        {
            try
            {
                using (openFileDialog)
                {
                    openFileDialog.FileName = String.Empty;
                    if (openFileDialog.ShowDialog() != DialogResult.OK)
                    {
                        return;
                    }

                    try
                    {
                        using (var fileStream = File.OpenRead(openFileDialog.FileName))
                        {
                            TextReader textReader = new StreamReader(fileStream);

                            sourceCodeTextBox.Text =
this.RemoveUnnecessaryCharacters(textReader.ReadToEnd());
                            for (var lineCount = 0; lineCount < sourceCodeTextBox.Lines.Length;
lineCount++)
                            {
                                var currentLine = sourceCodeTextBox.Lines[lineCount];
                                var nextLine = sourceCodeTextBox.Lines[lineCount + 1];
                                if ((String.IsNullOrEmpty(currentLine) ||
String.IsNullOrWhiteSpace(currentLine))
                                    && (String.IsNullOrEmpty(nextLine) ||
String.IsNullOrWhiteSpace(nextLine)))
                                {
                                    this._sourceCode = Environment.NewLine;
                                }
                                else
                                {
                                    this._sourceCode += currentLine + Environment.NewLine;
                                }
                            }

                            sourceCodeTextBox.Text = this._sourceCode;

                            textReader.Close();
                            fileStream.Close();
                        }
                    }
                    catch (Exception) { }
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show(exception.Message);
            }
        }

        public static void SaveToFile(SaveFileDialog saveFileDialog, TextBox sourceCodeTextBox)
        {
            try
            {
                saveFileDialog.FileName = String.Empty;
                using (saveFileDialog)
                {
                    if (saveFileDialog.ShowDialog() != DialogResult.OK)
                    {
                        return;
                    }

                    try
                    {
                        using (var fileStream = File.Create(saveFileDialog.FileName))
                        {
                            TextWriter textWriter = new StreamWriter(fileStream);

                            textWriter.Write(sourceCodeTextBox.Text);

                            textWriter.Close();
                            fileStream.Close();
                        }
                    }
                    catch (Exception exception)
                    {
                        MessageBox.Show(exception.Message);
                    }
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show(exception.Message);
            }
        }
    }
}
```

CSharpMcCabeHalstead.cs

```
using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;


namespace CSharpMcCabeHalstead
{
    public partial class CSharpMcCabeHalstead : Form
    {
        private readonly SourceCodeEditor _sourceCodeEditor = new SourceCodeEditor();
        private const string CyclomaticComplexityText = "The cyclomatic complexity of a program: ";
        private const string TotalNumberOfOperatorsText = "The total number of operators: ";
        private const string TotalNumberOfOperandsText = "The total number of operands: ";
        private const string NumberOfDistinctOperatorsText = "The number of distinct operators: ";
        private const string NumberOfDistrinctOperandsText = "The number of distinct operands: ";
        private const string ProgramVocabularyText = "The program vocabulary: ";
        private const string ProgramLengthText = "The program length: ";
        private const string ProgramVolumeText = "The program volume: ";
        private const string ProgramdifficultyText = "The program difficulty: ";
        private const string ProgrameffortText = "The program effort: ";
        private const string ProgramerrorText = "The program error: ";

        public static int N_operator;
        public static int N_operand;
        public static int D_operator;
        public static int D_operand;

        public CSharpMcCabeHalstead()
        {
            this.InitializeComponent();

            var sourceCode = File.ReadAllText(@"F:\VSA Help\csharp-mccabe-halstead-master\csharp-mccabe-
halstead-master\CSharpMcCabeHalstead\Program.cs") + Environment.NewLine;
            sourceCode += File.ReadAllText(@"F:\VSA Help\csharp-mccabe-halstead-master\csharp-mccabe-
halstead-master\CSharpMcCabeHalstead\CSharpMcCabeHalstead.cs") + Environment.NewLine;
            sourceCode += File.ReadAllText(@"F:\VSA Help\csharp-mccabe-halstead-master\csharp-mccabe-
halstead-master\CSharpMcCabeHalstead\SourceCodeEditor.cs") + Environment.NewLine;
            sourceCode += File.ReadAllText(@"F:\VSA Help\csharp-mccabe-halstead-master\csharp-mccabe-
halstead-master\CSharpMcCabeHalstead\McCabe.cs") + Environment.NewLine;
            sourceCode += File.ReadAllText(@"F:\VSA Help\csharp-mccabe-halstead-master\csharp-mccabe-
halstead-master\CSharpMcCabeHalstead\Halstead.cs");

            //this.sourceCodeTextBox.Text =
this._sourceCodeEditor.RemoveUnnecessaryCharacters(sourceCode);
            if (sourceCodeTextBox.Text == "")
            {
                PlotGraph.Visible = false;
            }
            else {
                PlotGraph.Visible = true;
            }

        }

        private void ClearForm()
        {
            this.cyclomaticComplexityLabel.Text = CyclomaticComplexityText;
            this.totalNumberOfOperatorsLabel.Text = TotalNumberOfOperatorsText;
            this.totalNumberOfOperandsLabel.Text = TotalNumberOfOperandsText;
            this.numberOfDistinctOperatorsLabel.Text = NumberOfDistinctOperatorsText;
            this.numberOfDistinctOperandsLabel.Text = NumberOfDistrinctOperandsText;
            this.programVocabularyLabel.Text = ProgramVocabularyText;
            this.programLengthLabel.Text = ProgramLengthText;
            this.programVolumeLabel.Text = ProgramVolumeText;
            this.programDifficultyLabel.Text = ProgramdifficultyText;
            this.programEffortLabel.Text = ProgrameffortText;
            this.programErrorLabel.Text = ProgramerrorText;

        }

        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.sourceCodeTextBox.Clear();
            this.ClearForm();
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this._sourceCodeEditor.OpenFromFile(this.openFileDialog, this.sourceCodeTextBox);
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SourceCodeEditor.SaveToFile(this.saveFileDialog, this.sourceCodeTextBox);
        }

        private void sourceCodeTextBox_DoubleClick(object sender, EventArgs e)
        {
            this._sourceCodeEditor.OpenFromFile(this.openFileDialog, this.sourceCodeTextBox);
        }

        private void mcCabeButton_Click(object sender, EventArgs e)
        {
            try {
                var mcCabe = new McCabe(this.sourceCodeTextBox.Text);
                this.cyclomaticComplexityLabel.Text = CyclomaticComplexityText
                    + Convert.ToString(mcCabe.CalculateCyclomaticComplexity());
            }
            catch (Exception ex)
            {
                MessageBox.Show("Please select a file.", "Error", MessageBoxButtons.OK);
            }

        }

        private void halsteadButton_Click(object sender, EventArgs e)
        {
            try {

                PlotGraph.Visible = true;

                var halstead = new Halstead(this.sourceCodeTextBox.Text);

                this.totalNumberOfOperatorsLabel.Text = TotalNumberOfOperatorsText
                    + Convert.ToString(halstead.CountTotalNumberOfOperators());

                this.numberOfDistinctOperatorsLabel.Text = NumberOfDistinctOperatorsText
                    + Convert.ToString(halstead.CountNumberOfDistinctOperators());

                this.totalNumberOfOperandsLabel.Text = TotalNumberOfOperandsText
                    + Convert.ToString(halstead.CountTotalNumberOfOperands());

                this.numberOfDistinctOperandsLabel.Text = NumberOfDistrinctOperandsText
                    + Convert.ToString(halstead.CountNumberOfDistinctOperands());

                this.programVocabularyLabel.Text = ProgramVocabularyText +
Convert.ToString(halstead.CalculateProgramVocabulary());

                this.programLengthLabel.Text = ProgramLengthText +
Convert.ToString(halstead.CalculateProgramLength());

                this.programVolumeLabel.Text = ProgramVolumeText +
Convert.ToString(halstead.CalculateProgramVolume());

                this.programDifficultyLabel.Text = ProgramdifficultyText +
Convert.ToString(halstead.CalculateProgramDifficulty());

                this.programEffortLabel.Text = ProgrameffortText +
Convert.ToString(Math.Round(halstead.CalculateProgramEffort()));

                this.programErrorLabel.Text = ProgramerrorText +
Convert.ToString(halstead.CalculateProgramErrors());


            }
            catch (Exception ex)
            {
                MessageBox.Show("Please select a file.", "Error", MessageBoxButtons.OK);
            }
        }

        private void sourceCodeTextBox_TextChanged(object sender, EventArgs e)
        {
            this.ClearForm();
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void FindDistinctOperatorPercentage(int val)
        {
            var halstead = new Halstead(this.sourceCodeTextBox.Text);

            decimal maxval = halstead.CountTotalNumberOfOperators();
            decimal minval = halstead.CountNumberOfDistinctOperators();

            val = Convert.ToInt32(minval);

            //decimal multi = (minval / maxval) * 100;

            //int percent = Convert.ToInt32(Math.Round(multi));

            //label1.Text = "" + minval;

        }


        private void PlotGraph_Click(object sender, EventArgs e)
        {
            plotGraph();
            PlotGraph.Visible = false;
        }

        private void plotGraph()
        {
            try {
                var halstead = new Halstead(this.sourceCodeTextBox.Text);

                decimal maxval = halstead.CountTotalNumberOfOperators();
                decimal minval = halstead.CountNumberOfDistinctOperators();

                barChart.Titles.Add("Halstead Plot");

                barChart.Series["Measure"].Points.AddXY("Total Operators", maxval);
                barChart.Series["Measure"].Points.AddXY("Total Operands",
halstead.CountTotalNumberOfOperands());
                barChart.Series["Measure"].Points.AddXY("Distinct Operators", minval);
                barChart.Series["Measure"].Points.AddXY("Distinct Operands",
halstead.CountNumberOfDistinctOperands());
                PlotGraph.Visible = false;

            }
            catch (Exception ex)
            {
                DialogResult dialogResult = MessageBox.Show(ex.Message + "/n", "Error",
MessageBoxButtons.OK);

            }

        }

        private void Refresh_Click(object sender, EventArgs e)
        {
            try {
                ClearForm();
                sourceCodeTextBox.Text = "";
                PlotGraph.Visible = false;
                barChart.Series.Clear();
                barChart.Titles.Clear();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message);
            }

        }

    }

}
```

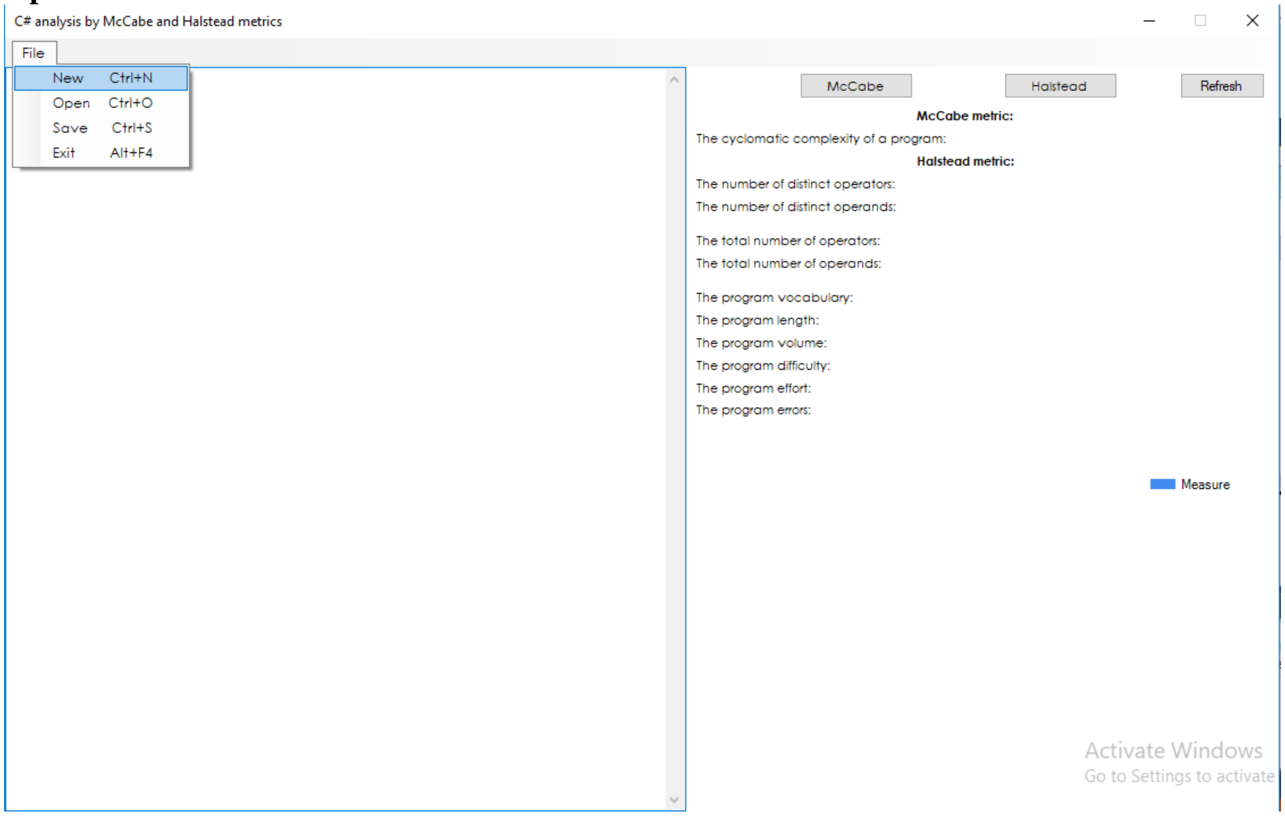
CSharpMcCabeHalstead.Designer.cs

```
namespace CSharpMcCabeHalstead
{
    partial class CSharpMcCabeHalstead
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (this.components != null))
            {
                this.components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea2 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend2 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series2 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            this.menuStrip = new System.Windows.Forms.MenuStrip();
            this.fileToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.newToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.openToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.saveToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.exitToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
            this.saveFileDialog = new System.Windows.Forms.SaveFileDialog();
            this.mcCabeButton = new System.Windows.Forms.Button();
            this.cyclomaticComplexityLabel = new System.Windows.Forms.Label();
            this.halsteadButton = new System.Windows.Forms.Button();
            this.totalNumberOfOperatorsLabel = new System.Windows.Forms.Label();
            this.numberOfDistinctOperatorsLabel = new System.Windows.Forms.Label();
            this.totalNumberOfOperandsLabel = new System.Windows.Forms.Label();
            this.numberOfDistinctOperandsLabel = new System.Windows.Forms.Label();
            this.programVocabularyLabel = new System.Windows.Forms.Label();
            this.programLengthLabel = new System.Windows.Forms.Label();
            this.programVolumeLabel = new System.Windows.Forms.Label();
            this.mcCabeMetricLabel = new System.Windows.Forms.Label();
            this.halsteadMetricLabel = new System.Windows.Forms.Label();
            this.programDifficultyLabel = new System.Windows.Forms.Label();
            this.programEffortLabel = new System.Windows.Forms.Label();
            this.programErrorLabel = new System.Windows.Forms.Label();
            this.PlotGraph = new System.Windows.Forms.Button();
            this.barChart = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.sourceCodeTextBox = new System.Windows.Forms.TextBox();
            this.Refresh = new System.Windows.Forms.Button();
            this.menuStrip.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.barChart)).BeginInit();
            this.SuspendLayout();
            //
            // menuStrip
            //
            this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.fileToolStripMenuItem});
            this.menuStrip.Location = new System.Drawing.Point(0, 0);
            this.menuStrip.Name = "menuStrip";
            this.menuStrip.Size = new System.Drawing.Size(1122, 25);
            this.menuStrip.TabIndex = 2;
            this.menuStrip.Text = "File";
            //
            // fileToolStripMenuItem
            //
            this.fileToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.newToolStripMenuItem,
            this.openToolStripMenuItem,
            this.saveToolStripMenuItem,
            this.exitToolStripMenuItem});
            this.fileToolStripMenuItem.Font = new System.Drawing.Font("Century Gothic", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.fileToolStripMenuItem.Name = "fileToolStripMenuItem";
            this.fileToolStripMenuItem.Size = new System.Drawing.Size(40, 21);
            this.fileToolStripMenuItem.Text = "File";
            //
            // newToolStripMenuItem
            //
            this.newToolStripMenuItem.Name = "newToolStripMenuItem";
            this.newToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Control | System.Windows.Forms.Keys.N)));
            this.newToolStripMenuItem.Size = new System.Drawing.Size(156, 22);
            this.newToolStripMenuItem.Text = "New";
            this.newToolStripMenuItem.Click += new System.EventHandler(this.newToolStripMenuItem_Click);
            //
            // openToolStripMenuItem
            //
            this.openToolStripMenuItem.Name = "openToolStripMenuItem";
            this.openToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Control | System.Windows.Forms.Keys.O)));
            this.openToolStripMenuItem.Size = new System.Drawing.Size(156, 22);
            this.openToolStripMenuItem.Text = "Open";
            this.openToolStripMenuItem.Click += new
System.EventHandler(this.openToolStripMenuItem_Click);
            //
            // saveToolStripMenuItem
            //
            this.saveToolStripMenuItem.Name = "saveToolStripMenuItem";
            this.saveToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Control | System.Windows.Forms.Keys.S)));
            this.saveToolStripMenuItem.Size = new System.Drawing.Size(156, 22);
            this.saveToolStripMenuItem.Text = "Save";
            this.saveToolStripMenuItem.Click += new
System.EventHandler(this.saveToolStripMenuItem_Click);
            //
            // exitToolStripMenuItem
            //
            this.exitToolStripMenuItem.Name = "exitToolStripMenuItem";
            this.exitToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Alt | System.Windows.Forms.Keys.F4)));
            this.exitToolStripMenuItem.Size = new System.Drawing.Size(156, 22);
            this.exitToolStripMenuItem.Text = "Exit";
            this.exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
            //
            // openFileDialog
            //
            this.openFileDialog.Filter = "C# source file (*.cs)|*.cs";
            this.openFileDialog.InitialDirectory = "d:\\Education\\Subjects\\3 Курс\\5
Семестр\\Предметы\\МССвИР\\LP\\Лабораторная работа №2\\" +
    "LAB2\\RubyMcCabeHalstead\\";
            //
            // saveFileDialog
            //
            this.saveFileDialog.Filter = "C# source file (*.cs)|*.cs";
            this.saveFileDialog.InitialDirectory = "d:\\Education\\Subjects\\3 Курс\\5
Семестр\\Предметы\\МССвИР\\LP\\Лабораторная работа №2\\" +
    "LAB2\\RubyMcCabeHalstead\\";
            //
            // mcCabeButton
            //
            this.mcCabeButton.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.mcCabeButton.Location = new System.Drawing.Point(700, 30);
            this.mcCabeButton.Name = "mcCabeButton";
            this.mcCabeButton.Size = new System.Drawing.Size(100, 23);
            this.mcCabeButton.TabIndex = 3;
            this.mcCabeButton.Text = "McCabe";
            this.mcCabeButton.UseVisualStyleBackColor = true;
            this.mcCabeButton.Click += new System.EventHandler(this.mcCabeButton_Click);
            //
            // cyclomaticComplexityLabel
            //
            this.cyclomaticComplexityLabel.AutoSize = true;
            this.cyclomaticComplexityLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.cyclomaticComplexityLabel.Location = new System.Drawing.Point(606, 80);
            this.cyclomaticComplexityLabel.Name = "cyclomaticComplexityLabel";
            this.cyclomaticComplexityLabel.Size = new System.Drawing.Size(231, 16);
            this.cyclomaticComplexityLabel.TabIndex = 4;
            this.cyclomaticComplexityLabel.Text = "The cyclomatic complexity of a program: ";
            //
            // halsteadButton
            //
            this.halsteadButton.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.halsteadButton.Location = new System.Drawing.Point(880, 30);
            this.halsteadButton.Name = "halsteadButton";
            this.halsteadButton.Size = new System.Drawing.Size(100, 23);
            this.halsteadButton.TabIndex = 5;
            this.halsteadButton.Text = "Halstead";
            this.halsteadButton.UseVisualStyleBackColor = true;
            this.halsteadButton.Click += new System.EventHandler(this.halsteadButton_Click);
            //
            // totalNumberOfOperatorsLabel
            //
            this.totalNumberOfOperatorsLabel.AutoSize = true;
            this.totalNumberOfOperatorsLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.totalNumberOfOperatorsLabel.Location = new System.Drawing.Point(606, 170);
            this.totalNumberOfOperatorsLabel.Name = "totalNumberOfOperatorsLabel";
            this.totalNumberOfOperatorsLabel.Size = new System.Drawing.Size(173, 16);
            this.totalNumberOfOperatorsLabel.TabIndex = 6;
            this.totalNumberOfOperatorsLabel.Text = "The total number of operators: ";
            //
            // numberOfDistinctOperatorsLabel
            //
            this.numberOfDistinctOperatorsLabel.AutoSize = true;
            this.numberOfDistinctOperatorsLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.numberOfDistinctOperatorsLabel.Location = new System.Drawing.Point(606, 120);
            this.numberOfDistinctOperatorsLabel.Name = "numberOfDistinctOperatorsLabel";
            this.numberOfDistinctOperatorsLabel.Size = new System.Drawing.Size(186, 16);
            this.numberOfDistinctOperatorsLabel.TabIndex = 7;
            this.numberOfDistinctOperatorsLabel.Text = "The number of distinct operators: ";
            //
            // totalNumberOfOperandsLabel
            //
            this.totalNumberOfOperandsLabel.AutoSize = true;
            this.totalNumberOfOperandsLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.totalNumberOfOperandsLabel.Location = new System.Drawing.Point(606, 190);
            this.totalNumberOfOperandsLabel.Name = "totalNumberOfOperandsLabel";
            this.totalNumberOfOperandsLabel.Size = new System.Drawing.Size(171, 16);
            this.totalNumberOfOperandsLabel.TabIndex = 8;
            this.totalNumberOfOperandsLabel.Text = "The total number of operands:";
            //
            // numberOfDistinctOperandsLabel
            //
            this.numberOfDistinctOperandsLabel.AutoSize = true;
            this.numberOfDistinctOperandsLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.numberOfDistinctOperandsLabel.Location = new System.Drawing.Point(606, 140);
            this.numberOfDistinctOperandsLabel.Name = "numberOfDistinctOperandsLabel";
            this.numberOfDistinctOperandsLabel.Size = new System.Drawing.Size(187, 16);
            this.numberOfDistinctOperandsLabel.TabIndex = 9;
            this.numberOfDistinctOperandsLabel.Text = "The number of distinct operands: ";
            //
            // programVocabularyLabel
            //
            this.programVocabularyLabel.AutoSize = true;
            this.programVocabularyLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programVocabularyLabel.Location = new System.Drawing.Point(606, 220);
            this.programVocabularyLabel.Name = "programVocabularyLabel";
            this.programVocabularyLabel.Size = new System.Drawing.Size(149, 16);
            this.programVocabularyLabel.TabIndex = 11;
            this.programVocabularyLabel.Text = "The program vocabulary: ";
            //
            // programLengthLabel
            //
            this.programLengthLabel.AutoSize = true;
            this.programLengthLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programLengthLabel.Location = new System.Drawing.Point(606, 240);
            this.programLengthLabel.Name = "programLengthLabel";
            this.programLengthLabel.Size = new System.Drawing.Size(119, 16);
            this.programLengthLabel.TabIndex = 12;
            this.programLengthLabel.Text = "The program length: ";
            //
            // programVolumeLabel
            //
            this.programVolumeLabel.AutoSize = true;
            this.programVolumeLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programVolumeLabel.Location = new System.Drawing.Point(606, 260);
            this.programVolumeLabel.Name = "programVolumeLabel";
            this.programVolumeLabel.Size = new System.Drawing.Size(126, 16);
            this.programVolumeLabel.TabIndex = 13;
            this.programVolumeLabel.Text = "The program volume: ";
            //
            // mcCabeMetricLabel
            //
            this.mcCabeMetricLabel.AutoSize = true;
            this.mcCabeMetricLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.mcCabeMetricLabel.Location = new System.Drawing.Point(800, 60);
            this.mcCabeMetricLabel.Name = "mcCabeMetricLabel";
            this.mcCabeMetricLabel.Size = new System.Drawing.Size(92, 15);
            this.mcCabeMetricLabel.TabIndex = 14;
            this.mcCabeMetricLabel.Text = "McCabe metric:";
            //
            // halsteadMetricLabel
            //
            this.halsteadMetricLabel.AutoSize = true;
            this.halsteadMetricLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.halsteadMetricLabel.Location = new System.Drawing.Point(800, 100);
            this.halsteadMetricLabel.Name = "halsteadMetricLabel";
            this.halsteadMetricLabel.Size = new System.Drawing.Size(93, 15);
            this.halsteadMetricLabel.TabIndex = 15;
            this.halsteadMetricLabel.Text = "Halstead metric:";
            //
            // programDifficultyLabel
            //
            this.programDifficultyLabel.AutoSize = true;
            this.programDifficultyLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programDifficultyLabel.Location = new System.Drawing.Point(606, 280);
            this.programDifficultyLabel.Name = "programDifficultyLabel";
            this.programDifficultyLabel.Size = new System.Drawing.Size(129, 16);
            this.programDifficultyLabel.TabIndex = 23;
            this.programDifficultyLabel.Text = "The program difficulty: ";
            //
            // programEffortLabel
            //
            this.programEffortLabel.AutoSize = true;
            this.programEffortLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programEffortLabel.Location = new System.Drawing.Point(606, 300);
            this.programEffortLabel.Name = "programEffortLabel";
            this.programEffortLabel.Size = new System.Drawing.Size(112, 16);
            this.programEffortLabel.TabIndex = 24;
            this.programEffortLabel.Text = "The program effort: ";
            //
            // programErrorLabel
            //
            this.programErrorLabel.AutoSize = true;
            this.programErrorLabel.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.programErrorLabel.Location = new System.Drawing.Point(606, 319);
            this.programErrorLabel.Name = "programErrorLabel";
            this.programErrorLabel.Size = new System.Drawing.Size(112, 16);
            this.programErrorLabel.TabIndex = 25;
            this.programErrorLabel.Text = "The program errors: ";
            //
            // PlotGraph
            //
            this.PlotGraph.Font = new System.Drawing.Font("Century Gothic", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.PlotGraph.Location = new System.Drawing.Point(983, 343);
            this.PlotGraph.Name = "PlotGraph";
            this.PlotGraph.Size = new System.Drawing.Size(100, 23);
            this.PlotGraph.TabIndex = 27;
            this.PlotGraph.Text = "Plot";
            this.PlotGraph.UseVisualStyleBackColor = true;
            this.PlotGraph.Click += new System.EventHandler(this.PlotGraph_Click);
            //
            // barChart
            //
            chartArea2.Name = "ChartArea1";
            this.barChart.ChartAreas.Add(chartArea2);
            legend2.Name = "Legend1";
            this.barChart.Legends.Add(legend2);
            this.barChart.Location = new System.Drawing.Point(609, 372);
            this.barChart.Name = "barChart";
            series2.ChartArea = "ChartArea1";
            series2.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;
            series2.Legend = "Legend1";
            series2.Name = "Measure";
            this.barChart.Series.Add(series2);
            this.barChart.Size = new System.Drawing.Size(501, 297);
            this.barChart.TabIndex = 29;
            this.barChart.Text = "chart1";
            //
            // sourceCodeTextBox
            //
            this.sourceCodeTextBox.Font = new System.Drawing.Font("Consolas", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.sourceCodeTextBox.Location = new System.Drawing.Point(0, 25);
            this.sourceCodeTextBox.MaxLength = 2147483647;
            this.sourceCodeTextBox.Multiline = true;
            this.sourceCodeTextBox.Name = "sourceCodeTextBox";
            this.sourceCodeTextBox.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.sourceCodeTextBox.Size = new System.Drawing.Size(600, 656);
            this.sourceCodeTextBox.TabIndex = 1;
            this.sourceCodeTextBox.TextChanged += new
System.EventHandler(this.sourceCodeTextBox_TextChanged);
            this.sourceCodeTextBox.DoubleClick += new
System.EventHandler(this.sourceCodeTextBox_DoubleClick);
            //
            // Refresh
            //
            this.Refresh.Location = new System.Drawing.Point(1035, 30);
            this.Refresh.Name = "Refresh";
            this.Refresh.Size = new System.Drawing.Size(75, 23);
            this.Refresh.TabIndex = 30;
            this.Refresh.Text = "Refresh";
            this.Refresh.UseVisualStyleBackColor = true;
            this.Refresh.Click += new System.EventHandler(this.Refresh_Click);
            //
            // CSharpMcCabeHalstead
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Center;
            this.ClientSize = new System.Drawing.Size(1122, 681);
            this.Controls.Add(this.Refresh);
            this.Controls.Add(this.barChart);
            this.Controls.Add(this.PlotGraph);
            this.Controls.Add(this.programErrorLabel);
            this.Controls.Add(this.programEffortLabel);
            this.Controls.Add(this.programDifficultyLabel);
            this.Controls.Add(this.halsteadMetricLabel);
            this.Controls.Add(this.mcCabeMetricLabel);
            this.Controls.Add(this.programVolumeLabel);
            this.Controls.Add(this.programLengthLabel);
            this.Controls.Add(this.programVocabularyLabel);
            this.Controls.Add(this.numberOfDistinctOperandsLabel);
            this.Controls.Add(this.totalNumberOfOperandsLabel);
            this.Controls.Add(this.numberOfDistinctOperatorsLabel);
            this.Controls.Add(this.totalNumberOfOperatorsLabel);
            this.Controls.Add(this.halsteadButton);
            this.Controls.Add(this.cyclomaticComplexityLabel);
            this.Controls.Add(this.mcCabeButton);
            this.Controls.Add(this.sourceCodeTextBox);
            this.Controls.Add(this.menuStrip);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.MainMenuStrip = this.menuStrip;
            this.MaximizeBox = false;
            this.Name = "CSharpMcCabeHalstead";
            this.ShowIcon = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "C# analysis by McCabe and Halstead metrics";
            this.menuStrip.ResumeLayout(false);
            this.menuStrip.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.barChart)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.MenuStrip menuStrip;
        private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem newToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem openToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem saveToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
        private System.Windows.Forms.OpenFileDialog openFileDialog;
        private System.Windows.Forms.SaveFileDialog saveFileDialog;
        private System.Windows.Forms.Button mcCabeButton;
        private System.Windows.Forms.Label cyclomaticComplexityLabel;
        private System.Windows.Forms.Button halsteadButton;
        private System.Windows.Forms.Label totalNumberOfOperatorsLabel;
        private System.Windows.Forms.Label numberOfDistinctOperatorsLabel;
        private System.Windows.Forms.Label totalNumberOfOperandsLabel;
        private System.Windows.Forms.Label numberOfDistinctOperandsLabel;
        private System.Windows.Forms.Label programVocabularyLabel;
        private System.Windows.Forms.Label programLengthLabel;
        private System.Windows.Forms.Label programVolumeLabel;
        private System.Windows.Forms.Label mcCabeMetricLabel;
        private System.Windows.Forms.Label halsteadMetricLabel;
        private System.Windows.Forms.Label programDifficultyLabel;
        private System.Windows.Forms.Label programEffortLabel;
        private System.Windows.Forms.Label programErrorLabel;
        private System.Windows.Forms.Button PlotGraph;
        private System.Windows.Forms.DataVisualization.Charting.Chart barChart;
        private System.Windows.Forms.TextBox sourceCodeTextBox;
        private System.Windows.Forms.Button Refresh;

    }
}
```

Project GUI:

Open File:



McCabe metric:

The cyclomatic complexity of a program:

Halstead metric:

The number of distinct operators:

The number of distinct operands:

The total number of operators:

The total number of operands:

The program vocabulary:

The program length:

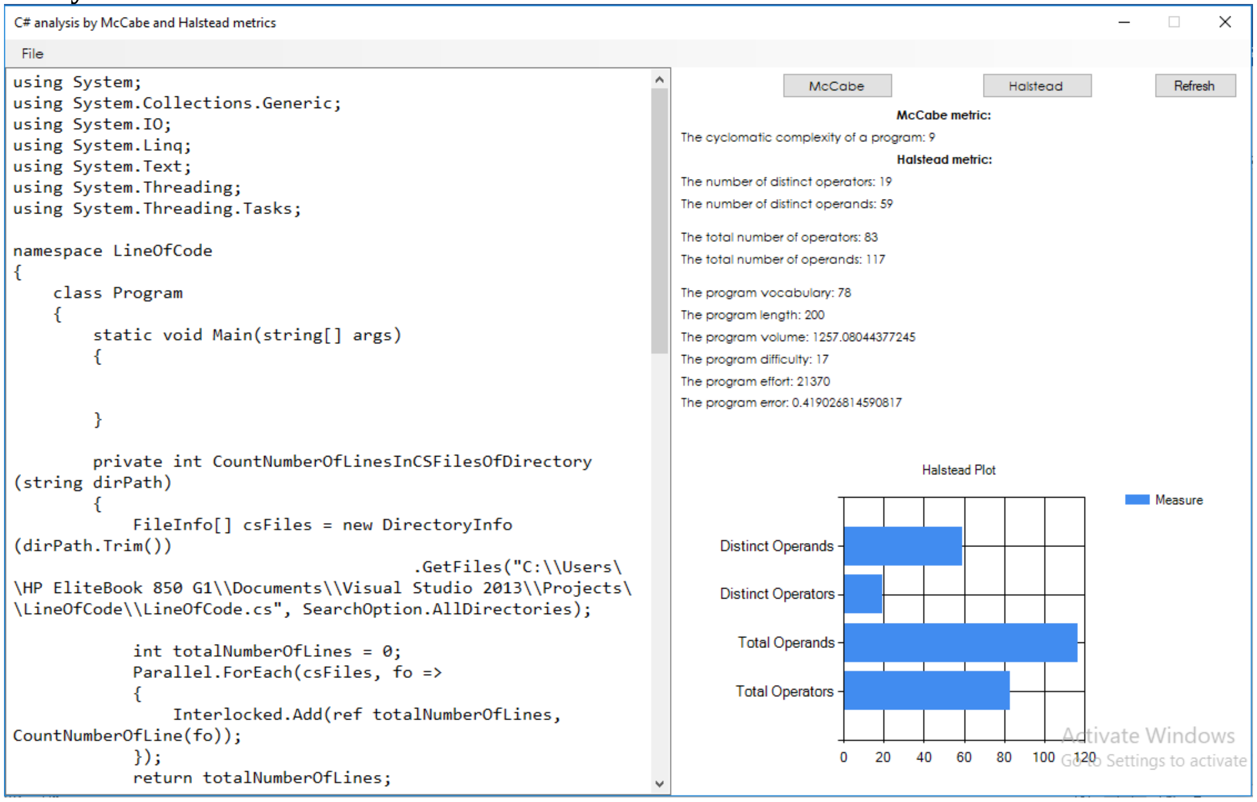
The program volume:

The program difficulty:

The program effort:

The program errors:

Analysis:



```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace LineOfCode
{
    class Program
    {
        static void Main(string[] args)
        {


        }

        private int CountNumberOfLinesInCSFilesOfDirectory
(string dirPath)
        {
            FileInfo[] csFiles = new DirectoryInfo
(dirPath.Trim())
                                        .GetFiles("C:\\Users\
\HP EliteBook 850 G1\\Documents\\Visual Studio 2013\\Projects\
\LineOfCode\\LineOfCode.cs", SearchOption.AllDirectories);

            int totalNumberOfLines = 0;
            Parallel.ForEach(csFiles, fo =>
            {
                Interlocked.Add(ref totalNumberOfLines,
CountNumberOfLine(fo));
            });
            return totalNumberOfLines;
```

McCabe metric:

The cyclomatic complexity of a program: 9

Halstead metric:

The number of distinct operators: 19

The number of distinct operands: 59

The total number of operators: 83

The total number of operands: 117

The program vocabulary: 78

The program length: 200

The program volume: 1257.08044377245

The program difficulty: 17

The program effort: 21370

The program error: 0.419026814590817