



QF205 Computing Technology For Finance

Project Report

# **Python Programming and its Applications in Income Tax Calculation**

G2 Team 9

Shan Huijie (01362618)

Xiao Tianyu (01370319)

Zhou Jingyi (01368193)

Zhang Yiling (01371404)

Zhang Zhengnan (01362557)

14 November 2020

<b>1. Introduction</b>	<b>4</b>
1.1 Income Tax Calculation	4
1.2 What is Programming Language	5
1.3 What is Python	5
<b>2. Setting Up Environment</b>	<b>5</b>
2.1 Installing IDE	5
2.2 Installing QtDesigner	6
2.3 Installing Visual Studios Code	8
<b>3. Overview of the Application</b>	<b>9</b>
3.1 Basic User Version	9
3.2 Gold User Version	10
<b>4. Basic User: Introduction to Python</b>	<b>11</b>
4.1 Variables and Variable Assignments	11
4.2 Data Types	12
4.2.1 Overview of Data Type	12
4.2.2 Numeric Literals	13
4.2.3 Sequence Types	14
4.2.4 Boolean	15
4.3 Arithmetic Operators	15
4.4 Built-in function	16
4.4.1 Iterators and Iterables	16
4.4.2 Lambda Function	17
4.4.3 Filter()	17
4.4.4 Next()	17
4.4.5 Enumerate()	17
4.4.6 Append()	18
4.4.7 input()	18
4.4.8 int()	18
4.4.9 float()	19
4.4.10 replace()	19
4.4.11 split ()	19
4.4.12 Applications in our application	19
4.5 If-Else Statements	20
<b>5. Gold User: Advanced Python</b>	<b>21</b>
5.1 Class	21
5.1.1 Attribute	22
5.1.2 Method	23
5.2 Pandas	24
5.2.1 Retrieve HTML	24
5.2.2 DataFrame	24
5.3 Graphical User Interface (GUI)	25
5.3.1 Interface Design	25

5.3.2 Import Statements	28
5.3.3 Link GUI to Python	29
<b>6. Appendix</b>	<b>30</b>
6.1 Code for Basic User	30
6.2 Code for Premium User	31
6.2.1 Tax.py	31
6.2.2 Tax_calculation.py	32
6.2.3 IncomeTaxCalculator.ui	34
<b>7. References</b>	<b>38</b>

# 1. Introduction

## 1.1 Income Tax Calculation

You are an employee of Singapore Management University, you are preparing for individual income tax for AY2020.

Back in university, you learned how to calculate individual income tax using Python in QF205. The Resident Income Tax Rate follows the instructions shown in the website of the Inland Revenue Authority of Singapore. The details are shown below:

From YA 2017 onwards

Chargeable Income	Income Tax Rate (%)	Gross Tax Payable (\$)
First \$20,000 Next \$10,000	0 2	0 200
First \$30,000 Next \$10,000	- 3.50	200 350
First \$40,000 Next \$40,000	- 7	550 2,800
First \$80,000 Next \$40,000	- 11.5	3,350 4,600
First \$120,000 Next \$40,000	- 15	7,950 6,000
First \$160,000 Next \$40,000	- 18	13,950 7,200
First \$200,000 Next \$40,000	- 19	21,150 7,600
First \$240,000 Next \$40,000	- 19.5	28,750 7,800
First \$280,000 Next \$40,000	- 20	36,550 8,000
First \$320,000 In excess of \$320,000	- 22	44,550

(IRAS, 2020)

## **1.2 What is Programming Language**

A programming language is a formal language consisting of a set of instructions used to make a computer execute certain tasks.

There are many programming languages now, some examples include Python, Java, R... For each of them, there are different syntaxes and keywords.

## **1.3 What is Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. (Python, n.d.) Python also has easy to understand and easy to use syntaxes, making it a good language to learn and use.

## **2. Setting Up Environment**

### **2.1 Installing IDE**

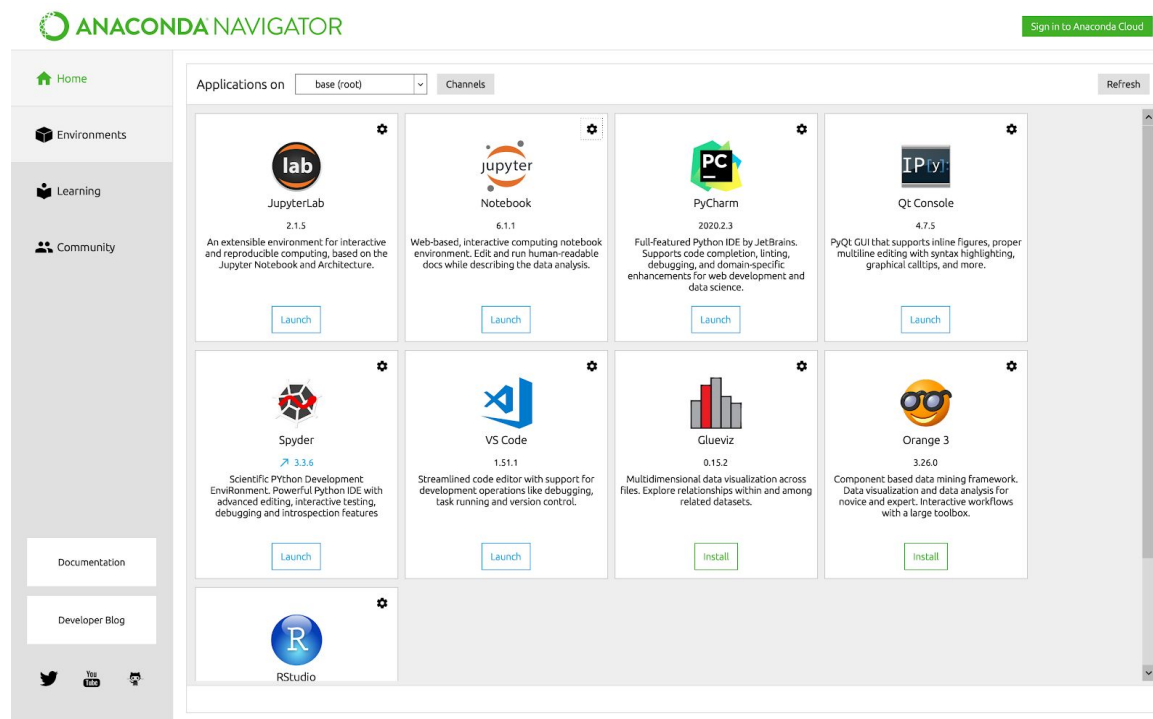
IDE stands for integrated development environment, which is where you could consolidate the different aspects of writing a computer program.

MacOS and Linux uses Terminal to run Python programs. For Windows users, You need to download and install Anaconda Prompt 2019.03 (Python 3.7) and Qt 5.6.0.

There is not only one way to use python on your computer and Anaconda is one of the more convenient ways to install and manage your python. Anaconda is a free distribution of python, which is a bundle of some popular Python packages and a package manager called conda. In order to operate python programs, all relevant packages must be installed beforehand. So it's recommended to use Anaconda rather than only Python.

Jupyter notebook is an open-source web application that allows the creation of live codes, equations, visualisations and narrative text. Unlike other applications like sublime text, Jupyter Notebook is able to show the outputs of each line immediately, making it an extremely useful learning tool for beginners.

To install jupyter notebook, access the Anaconda Navigator which we installed earlier.



Click “Launch” Button below Jupyter Notebook to start the installation. After installation, you can also click on the same button to launch Jupyter Notebook and start coding.

## 2.2 Installing QtDesigner

Qt Designer is the Qt tool for designing and building graphical user interfaces. It allows you to design widgets, dialogs or complete main windows using on-screen forms and a simple drag-and-drop interface. It has the ability to preview your designs to ensure they work as you intended, and to allow you to prototype them with your users, before you have to write any code.

For mac Users, click on this [link](https://build-system.fman.io/static/public/files/Qt%20Designer.dmg) or paste the following URL to your web browser.

<https://build-system.fman.io/static/public/files/Qt%20Designer.dmg>

For Windows Users, click on this [link](#) or paste the following URL to your web browser.

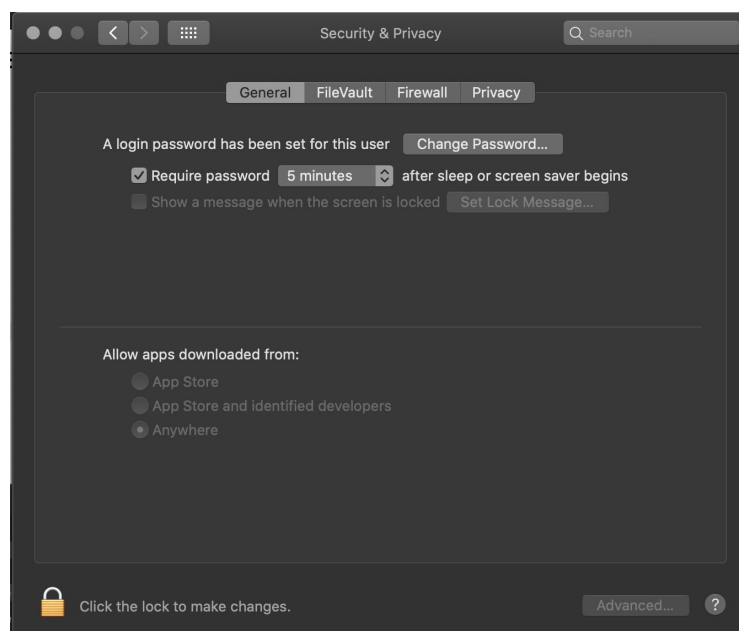
<https://build-system.fman.io/static/public/files/Qt%20Designer%20Setup.exe>

If you want to further information about setting up Qt, please click this [link](#).

Skip the following section if you are not using macOS during the installation of QtDesigner.

### *Step 1:*

Open Systems Preferences > Security & Privacy > General. There is a section on allowing applications to be downloaded from either App Store, or App Store and identified developers.

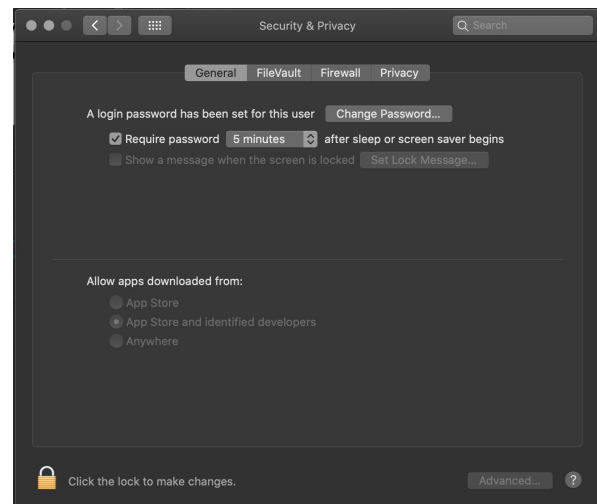
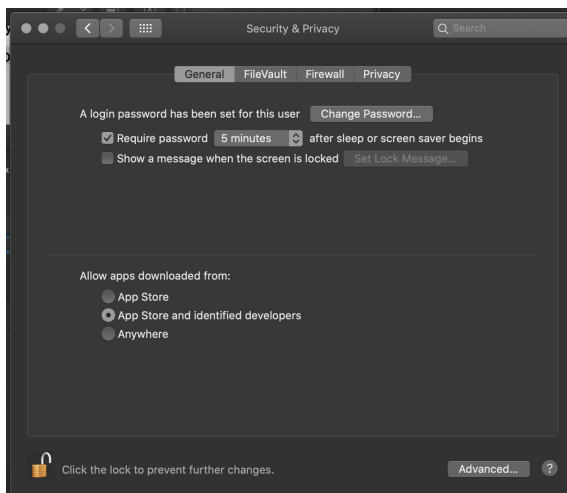


### *Step 2:*

Change the systems Preferences to successfully download applications from an external source. Please follow the following steps:

Click on the padlock with the label-> click the lock to make changes-> enter password to give permission to update the system preferences setting-> Change the

settings to 'App Store and identified developers' ->click the padlock again to confirm the update.



## 2.3 Installing Visual Studio Code

We recommend using Microsoft's Visual Studio Code (VS Code) for the integration of GUI. VS Code is the preferred IDE for the following benefits - The IDE is a free open-source editor compatible with Windows, Linux and macOS, and also it supports numerous major programming languages, such as Python, JavaScript, Java, and more.

To install and set the environment for VS Code, please click on this [link](https://code.visualstudio.com/Download) or copy the following URL to your web browser.

<https://code.visualstudio.com/Download>

To read up more detailed information on setting up VS Code, please follow the platform specific guides for [macOS](#), [Windows](#), and [Linux](#).



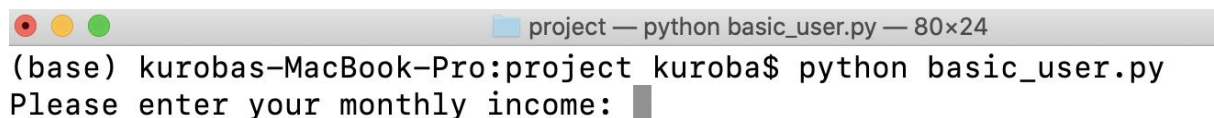
### 3. Overview of the Application

Our application is separated into the Basic User Version and the Gold User Version. The Basic User Version will be executed on the terminal of the user's computer, while the Gold User Version will have a GUI for the users to operate on. The following paragraphs in this function will introduce the two versions to you.

#### 3.1 Basic User Version

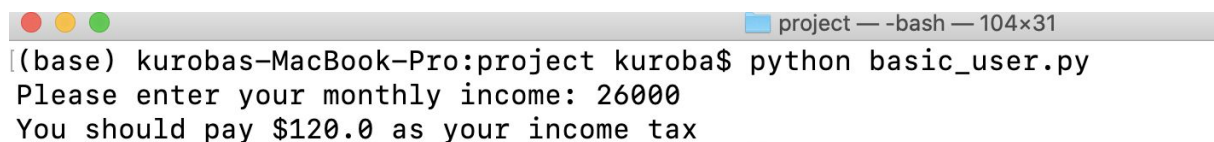
To execute the code of the basic user version, mac users need to open their terminal, windows users need to open their Anaconda Prompt. Then, you would need to use the command prompt to locate the python file (basic\_user.py). After that, key in **python basic\_user.py** to call the application.

The following screen should be seen:



```
project — python basic_user.py — 80x24
(base) kurobas-MacBook-Pro:project kuroba$ python basic_user.py
Please enter your monthly income: █
```

Then, input your income, and your chargeable tax would be displayed. Refer to the screenshot below to see an example.

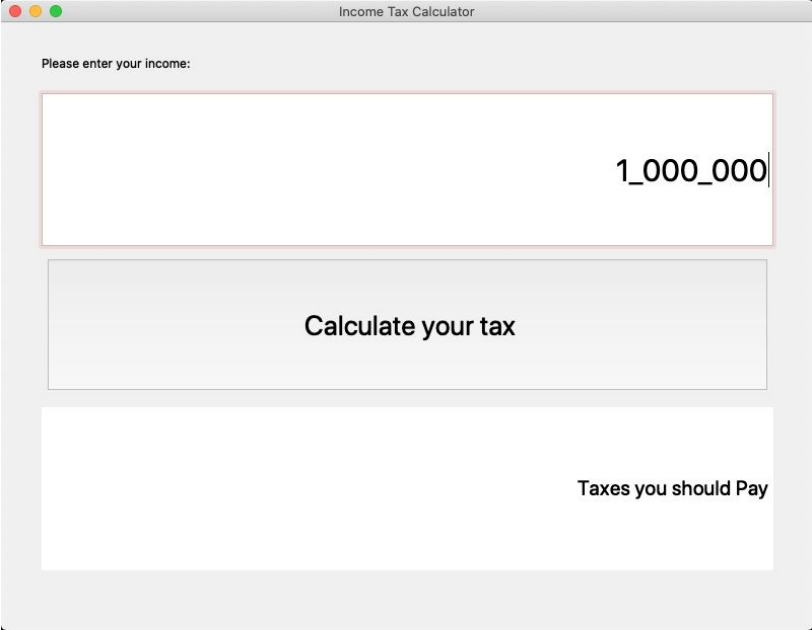


```
project — -bash — 104x31
[(base) kurobas-MacBook-Pro:project kuroba$ python basic_user.py
Please enter your monthly income: 26000
You should pay $120.0 as your income tax _
```

## 3.2 Gold User Version

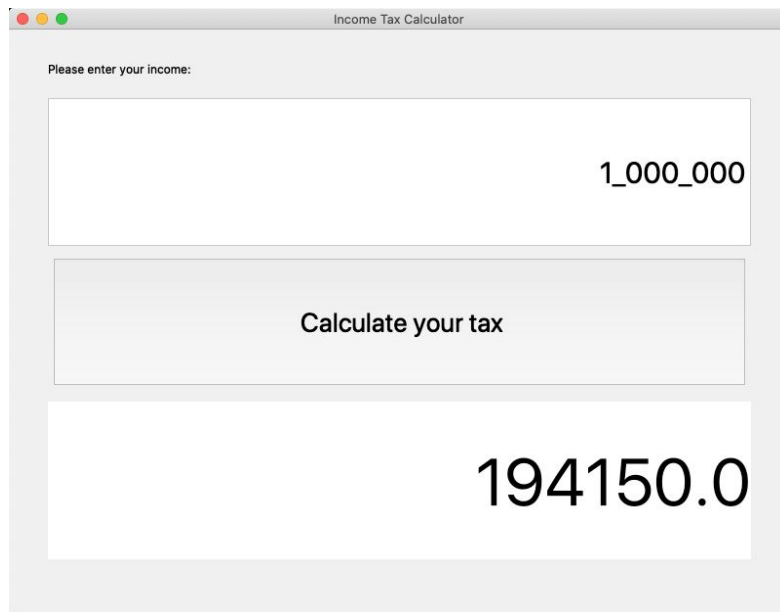
To execute the code of the gold user version, you also need to open your terminal or Anaconda Prompt. Then, use the command prompt to locate the python file (). After that, key in **python Tax\_calculation.py** to open the application.

The following window should be displayed when you execute the above command. You could enter your income into the first cell (now filled with 1\_000\_000), and press the 'Calculate your tax' button to make the application calculate your income tax.



The screenshot shows a window titled "Income Tax Calculator". Inside the window, there is a label "Please enter your income:" followed by a text input field containing the value "1\_000\_000". Below the input field is a button labeled "Calculate your tax". At the bottom of the window, there is a label "Taxes you should Pay" followed by a large empty rectangular box for the output.

As shown below, for example you have keyed in your income as \$1,000,000, and pressed the 'Calculate your tax' button, your income tax will be calculated and displayed in the cell at the bottom.



## 4. Basic User: Introduction to Python

### 4.1 Variables and Variable Assignments

To start with python, or even any programming language, we need to understand the concept of a variable. Variables store the assigned information that will be further used in a written program. In python, variables are just like containers for storing data values.

The process of storing a data object into a variable is called Variable Assignment. In python, it is a very simple process, done by using a '=' sign. It assigns the variable name to the left side of the '=' sign, and the value to be assigned on the right side of the '=' sign.

However, there are naming conventions that you need to follow when assigning variables. You should name your variables that best describe the data that is assigned to it. Also, you need to comply with the following rules:

1. A variable name can contain the following: Alphanumeric characters (A-Z, 0-9) and underscores (\_).
2. A variable name must start with a letter or underscore.
3. A variable cannot start with a number.
4. Variable names are case sensitive.

5. Do not use python keywords as variable names. Such keyword includes:

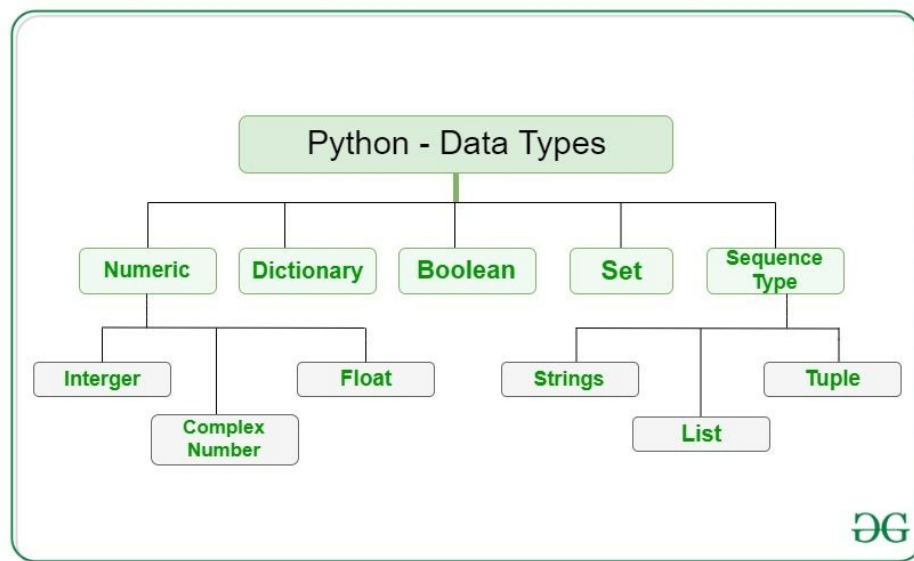
False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

In our case of calculating income taxes, we have used variable assignments in many ways. One example is assigning the tax rates to a variable named 'income\_tax\_rate'. See the following screenshot for more information.

```
income_tax_rate=[2.0, 3.5, 7.0, 11.5, 15.0, 18.0, 19.0, 19.5, 20.0, 22.0]
```

## 4.2 Data Types

### 4.2.1 Overview of Data Type



(source: Geeksforgeeks)

In order for us to calculate the income tax of a given income, we need to assign the variables to many kinds of data types. The picture above shows part of the data types in python. We will be going through some of the simple data types in the following sections.

### 4.2.2 Numeric Literals

Numeric literals consist of integers, floating point numbers, imaginary numbers. Numeric literals do not include a sign. For example, a phrase like '-1' is not a numeric literal which is an expression. It consists of an operator '-' and literal 1.

There is no limit on the length of integer literals apart from what can be stored in the available memory. Floating point numbers can be divided into 2 parts which are point-float (i.e 1.234) and exponent-float numbers (i.e 1234e10).

Underscores are allowed and are used to group digits for enhanced readability. They are ignored when determining the numeric value of the literal. However, it is only valid when one underscore occurs between digits and after base specifiers. For example, '7\_00\_00' is valid while '7\_\_011' is illegal.

Leading zeros in non-zero decimal numbers are not allowed. Decimals here refer to the base-10 integer and not just a number with decimal points. For example, '077e010' is valid while '077' is illegal. A reference of the bases is as shown in the table below.

Base	Description
Base-2 (Binary)	0 & 1
Base-8 (Octal)	0,1,2,3,4,5,6,7
Base-10 (Decimal)	0,1,2,3,4,5,6,7,8,9
Base-16 (Hexadecimal)	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

In the above section, we will show some examples of the numeric literals used in the basic user version and gold user version of the income tax calculator.

```
tax=0.0
```

Tax variable here is a numeric literal which is a floating point number. Above code shows the initialization of the tax variable.

### 4.2.3 Sequence Types

There are many sequence types variables in Python. We would be going through the simple ones that will be used in our application.

#### **Strings**

Strings in Python represents unicode characters. Strings can be enclosed in matching single quotes (') or double quotes ("). They can also be enclosed in matching groups of three single or double quotes (these are generally referred to as triple-quoted strings). The joining (concatenation) of strings can be done by using the '+' sign.

```
"Please enter your monthly income: "
```

The screenshot above represents a string that is enclosed in double quotes inside our code for the basic users.

#### **Lists**

List is a very important data type, and is used frequently in our application. List is a compound data type. It can be written as a list of comma-separated values, inside square brackets. Inside a list, there could be values of different data types, but in the common case, it only contains values of the same data type. Below is an example of the list used in our project.

```
gross_tax_payable=[0, 200, 550, 3350, 7950, 13950, 21150, 28750, 36550,
44550]
income_tax_rate=[2.0, 3.5, 7.0, 11.5, 15.0, 18.0, 19.0, 19.5, 20.0,
22.0]
chargeable_income=[20000, 30000, 40000, 80000, 120000, 160000, 200000,
240000, 280000, 320000]
```

Elements in a list have sequences, the first element has a sequence number of 0 and the others element follows an increasing sequence number.

#### 4.2.4 Boolean

Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations. Booleans are used to represent truth values (either True or False). One may employ the use of the built-in function `bool()` to cast any value to a Boolean.

```
if i==0:
```

In the case of the basic user version, booleans are used when doing simple comparisons as seen below. When `i` equals 0, the result will be true, otherwise, the result will be false.

### 4.3 Arithmetic Operators

In Python, arithmetic operations are used with numeric literals to perform mathematical operations. Below is a table of the arithmetic operators in python.

Operator	Description	Example
+ Addition	Adds the values on both sides of the operator	1 + 1 = 2
- Substraction	Subtracts right hand value from the left hand.	9 - 1 = 8

/ Division	Divides left hand operand by right hand operand, returning a float.	8 / 2 = 2.0
// Floor Division	The digits after the decimal point after the division is removed.	7 // 2 = 3
* Multiplication	Multiplies the values on either side of the operators	1 * 2 = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	8 % 6 = 2
** Exponent	Does exponential calculation on numbers.	2 ** 2 = 4

Now that you have gotten a basic idea of the arithmetic operations in python, let us look at some examples inside our code! Referring to the picture below (not a complete formula), we are doing an additional calculation between the two variables 'gross\_tax\_payable' and 'income\_tax\_rate'.

```
gross_tax_payable[i-1]+income_tax_rate[i-1]
```

## 4.4 Built-in function

### 4.4.1 Iterators and Iterables

An **iterator** is an object that can be iterated upon, meaning that you can traverse through all the values. Technically, in Python, an **iterator** is an object which implements the iterator protocol, which consists of the methods `__iter__()` and `__next__()`.



An **iterable** is any Python object capable of returning its members one at a time, permitting it to be iterated over in a for-loop. Familiar examples of **iterables** include lists, tuples, and strings - any such sequence can be iterated over in a for-loop. An iterable must be a sequence, an **iterator**, or some other object which supports iteration.

#### **4.4.2 Lambda Function**

In python, there exists a type of function called the 'Lambda Functions'. Lambda expressions (sometimes called lambda forms) are used to create anonymous functions. The expression 'lambda arguments: expression' yields a function object.

#### **4.4.3 Filter()**

The built-in function `filter(function, iterable)` returns an iterator from those elements of iterable for which function returns true. It takes in 2 parameters which are function and iterable. The parameter *function* is a function to be run for each item in the iterable

#### **4.4.4 Next()**

`Next(iterable, default)` is a built-in function which returns the next item in an iterator. It takes in 2 parameters which are *iterable* and *default*. *Iterable* is a required parameter while *default* is optional.

#### **4.4.5 Enumerate()**

`Enumerate(iterable, start=0)` is a built-in function which returns an enumerate object that is an iterator. The `__next__()` method of the iterator returned by

`enumerate()` returns a tuple containing a count (from *start* which defaults to 0) and the values obtained from iterating over *iterable*.

#### 4.4.6 Append()

`Append(element)` is used when you want to add an element at the back of a list. It contains one parameter *element* which is an element of any type (string, number, object etc.) In our case, we append the `math.inf` to the end of the `chargeable_income` list.

```
# Append math.inf to chargeable_income.  
chargeable_income.append(math.inf)
```

#### 4.4.7 input()

The function `input(prompt)` reads a line entered on a console by an input device such as a keyboard, and converts it to a string (stripping a trailing newline), and returns that. It takes in one parameter *prompt* which is a String, representing a default message before the input.

#### 4.4.8 int()

The function `int(x)` converts a specified value into an integer object. This function takes in a number or a string containing a number (e.g. '6'). Then it outputs the input as an integer value. See picture below for our implementation of the `int()` function in our codes. We converted the chargeable income an integer.

```
chargeable_income=[int(x.split()[1].replace('$','').replace(',','')) for x in IncomeTaxRate_Table['Chargeable Income'] ]
```

#### 4.4.9 float()

The function `float(x)` converts a specified value into a floating point number. This function can take in a number or a string containing a number (e.g. '3'). Then it outputs the input as a float value. See picture below for our implementation of the `float()` function, where we convert the income tax rate into a float.

```
income_tax_rate=[float(x.split()[1]) for x in IncomeTaxRate_Table['Income Tax Rate (%)']]
```

#### 4.4.10 replace()

The `replace(oldvalue, newvalue, count)` function replaces a specified phrase with another specified phrase. For example, given a text "one one was a race horse, two two was one too.", `txt.replace("one", "three")` will replace all the "one" with "three". The picture below shows how we apply `replace()` method.

```
int(x.split()[1].replace('$', ''))
```

In our codes, we remove the \$ sign by replacing the "\$" with nothing.

#### 4.4.11 split ()

The `split(sep=None, maxsplit=-1)` function splits a string into a list where each word is a list item. It will return a list of the words in the string, using `sep` as the delimiter string. If `sep` is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings (for example, `'1,,2'.split(',')` returns `['1', '', '2']`). If `sep` is not given, it will take consecutive whitespaces as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

In our case, we didn't specify the `sep` as the income tax rate is separated by empty spaces.

```
income_tax_rate=[float(x.split()[1]) for x in  
IncomeTaxRate_Table['Income Tax Rate (%)']]
```

#### 4.4.12 Applications in our application

In our case, we apply the enumerate function to the chargeable\_income variable which is a list mentioned above. The function returns a tuple containing a count (from start of 0) and the values obtained from the iterating over iterables. Some examples of this could be (0, 20000), (1, 30000) and so on.

Then we apply a lambda function to it which will compare the income with the values obtained from the iterating over iterables.

After that, we use the filter function to the lambda function and enumerate function. This filter function will return us with the iterable enumerate object that satisfies the lambda function's criterion.

The next function is then used to retrieve the next item from the filter function, which then finds the position of the first position greater than the chargeable income.

```
i=next(filter(lambda s: s[1]>income, enumerate(chargeable_income)))[0]
```

## 4.5 If-Else Statements

After getting the i from the previous function, which is used to determine the index of the gross tax payable, income tax rate and chargeable income, we would use an 'If-Else' Statement to determine the final amount of tax payable based on the income inputted.

An If-Else statement is a conditional statement, where there will be a condition after the If-Else clause. The functionality of it is very similar to how we use the word 'if' in our daily lives.

If the condition is satisfied after the 'if' clause, then the code will execute the actions in the next indented line. See the picture below, if i is equal to 0, then tax is 0.

On the other hand, if the condition is not satisfied after the 'if' clause, it will look for the 'else' clause, and execute the actions in the next indented line. In the picture

below, if i is not equals to 0, then tax will be calculated by the formula as shown in the picture.

```
# If i equals 0, tax=0
if i==0:
    tax=0.0
# If i is greater than 0), use gross_tax_payable[i-1], income_tax_rate[i-1] and chargeable_income[i-1]
# to compute y=gross_tax_payable[i-1]+income_tax_rate[i-1]/100*(x-chargeable_income[i-1])
else:
    tax=gross_tax_payable[i-1]+income_tax_rate[i-1]/100*(income-chargeable_income[i-1])
```

## 5. Gold User: Advanced Python

### 5.1 Class

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Many classes like to create objects with instances customized to a specific initial state. Therefore a class may define a special method `__init__`, also known as constructor. When a class defines an `__init__()` method, class instantiation automatically invokes `__init__()` for the newly-created class instance.

We define classes by using the class keyword, similar to how we define functions by using the def keyword. There are two classes built in this application, **Tax Class** in Tax.py and **Main Class** in Tax\_calculation.py. A simple example of how to define a class is shown below.

```
class Tax:
    def __init__(self, your_in
        self.gross_tax_payabl
```

### 5.1.1 Attribute

#### Main Class

```
class Main(QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.label_ChargableIncome.setText(
            "<span style='font-size:12pt;'"
            "Please enter your income: "
            "</span>"
        )
        self.pushButton_Calculate.clicked.connect(self.Tax_Calculator)
```

In the Main class, there are two attributes, QMainWindow and Ui\_MainWindow. The attributes are for initializing the GUI. A “\_\_init\_\_” function is defined. In this function, we need to setupUi, create the actual instance of widgets, and connect the pushButton with a function in the class.

The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class. Self parameter itself has no meaning and it does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class. Conventionally, we would use the name self in our case.

#### Tax Class

```
class Tax:
    def __init__(self, your_income, gross_tax_payable, income_tax_rate, chargeable_income):
        self.gross_tax_payable = gross_tax_payable
        self.income_tax_rate = income_tax_rate
        self.chargeable_income = chargeable_income
        self.your_income = your_income
```

In the Tax class, every instance of tax has four parameters including your\_income, gross\_tax\_payable, income\_tax\_rate, and chargeable\_income. Your\_income is the user input income in the GUI. The other three parameters are derived by reading the html and extracting from the tax rate table. The details of reading the html part will be explained below. Tax Class consists of the initialization and tax\_to\_pay functions.

### 5.1.2 Method

#### Main Class Method :Tax\_Calculator

```
def Tax_Calculator(self):
    income = float(self.lineEdit_Income.text())

    # Calculate Tax |
    your_tax = Tax(income, gross_tax_payable, income_tax_rate, chargeable_income)
    tax = your_tax.tax_to_pay()
    self.label_Tax.setText(str(tax))
```

This method is the main building block of the tax calculator application which first retrieves the income from the text input in the GUI. Then it will create an object of Tax class and call the Tax class method tax\_to\_pay to calculate the final tax required to pay. Lastly, it will display the final tax result in the GUI.

#### Tax Class Method: tax\_to\_pay

```
def tax_to_pay(self):
    import math
    self.chargeable_income.append(math.inf)
    # Find in xi, the location (i) of the first number which is greater than x,
    i = next(filter(lambda s: s[1] > self.your_income, enumerate(self.chargeable_income)))[0]
    # If i equals 0, tax=0
    if i == 0:
        tax = 0
    # If i is greater than 0, use bi[i-1], mi[i-1] and xi[i-1]
    # to compute y = bi[i-1] + mi[i-1]/100 * (x - xi[i-1])
    else:
        tax = self.gross_tax_payable[i-1] + self.income_tax_rate[i-1]/100 * (self.your_income - self.chargeable_income[i-1])

    return tax
```

This function is the main building block of Tax Class to calculate the respective tax to be paid according to the yearly income input by the user.

Firstly, this function will locate the first number in the list of chargeable\_income which is greater than the input yearly income. If the number is 0, suggesting that the input yearly income is lower than any chargeable income. If the number is greater than 0, this function will further look for respective income\_tax\_rate and gross\_tax\_payable to calculate the tax to be paid. Return statement will return the output generated by the function.

## 5.2 Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language (Pandas, n.d.). In our application, this module will contribute to retrieving the tax rate information by calling the HTML retrieve function.

### 5.2.1 Retrieve HTML

If there are any updates on Resident Income Tax in the website, for example, income tax rate, using a HTML retrieval function will be able to get the latest information about income tax. Hence, we use `pandas.read_html` to read HTML tables into a list of DataFrame objects. The result retrieved from the website of Inland Revenue Authority of Singapore for AY2017 onwards is as follows:

	Chargeable Income	Income Tax Rate (%)	Gross Tax Payable (\$)
0	First \$20,000 Next \$10,000	0 2	0 200
1	First \$30,000 Next \$10,000	- 3.50	200 350
2	First \$40,000 Next \$40,000	- 7	550 2,800
3	First \$80,000 Next \$40,000	- 11.5	3,350 4,600
4	First \$120,000 Next \$40,000	- 15	7,950 6,000
5	First \$160,000 Next \$40,000	- 18	13,950 7,200
6	First \$200,000 Next \$40,000	- 19	21,150 7,600
7	First \$240,000 Next \$40,000	- 19.5	28,750 7,800
8	First \$280,000 Next \$40,000	- 20	36,550 8,000
9	First \$320,000 In excess of \$320,000	- 22	44550

### 5.2.2 DataFrame

DataFrames (*data*, *index*, *columns*) in Pandas are two-dimensional, size-mutable, potentially heterogeneous tabular data.



Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. It can be thought of as a dict-like container for Series objects. The primary pandas data structure.

In order to retrieve specific column from dataframe, we can specify the column name as a string in the bracket.

```
IncomeTaxRate_Table['Income Tax Rate (%)']
```

### **5.3 Graphical User Interface (GUI)**

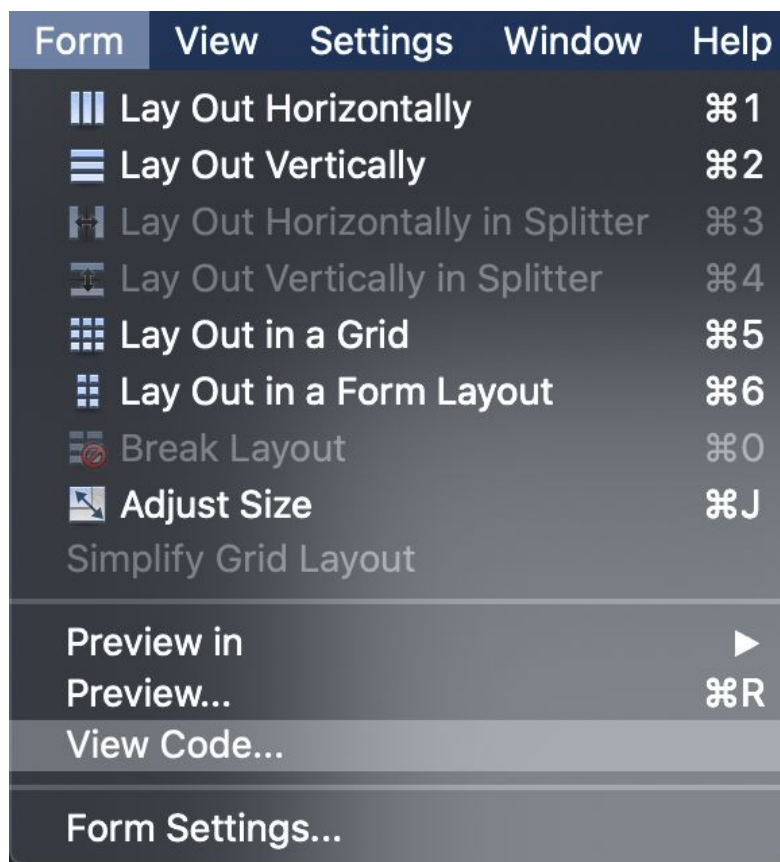
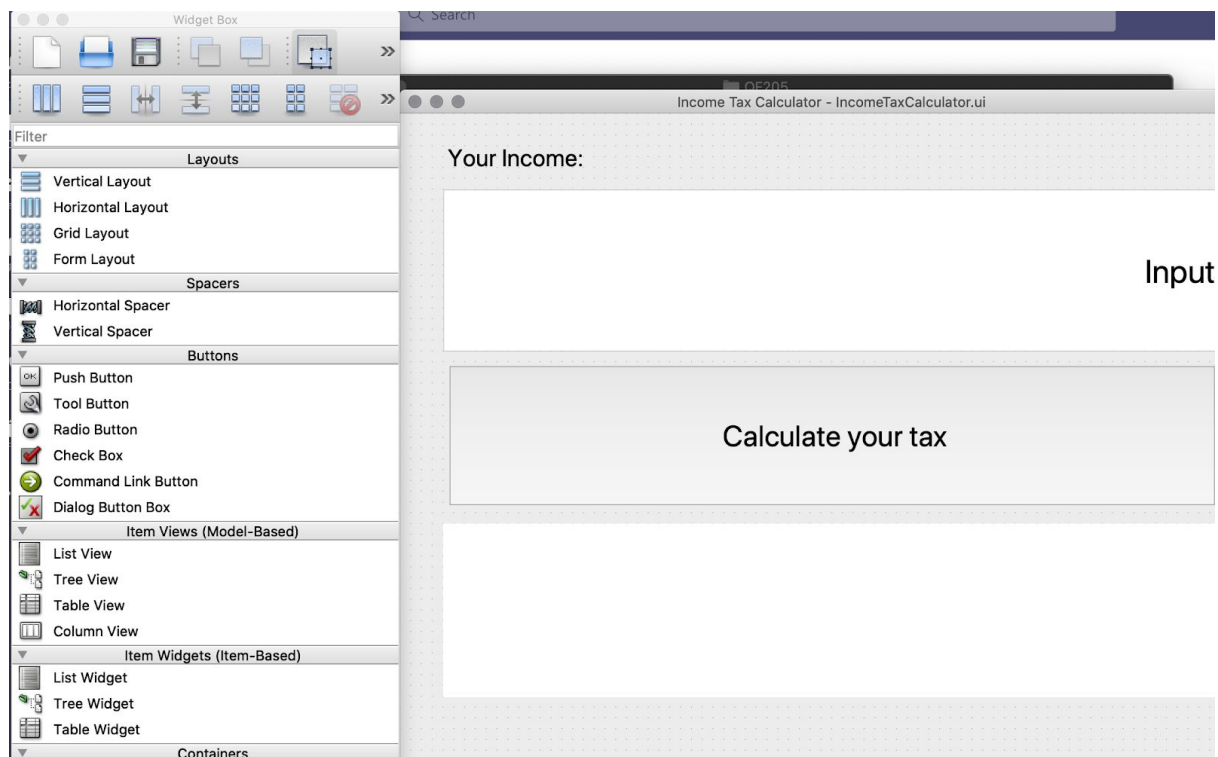
We have learned how to code for tax class, now we can start to build our application using GUI! GUI stands for graphical user interface. In the following tutorial, we will introduce how to QT designer to build the interface and link to our python code for our application.

#### **5.3.1 Interface Design**

Qt Designer is the Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. You can compose and customize your windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions.

To design your interface, you can simply choose the elements you need and drag them into the interface directly. In the figure below, you can see how to utilize the widget, rename the name attribute for each label, textbox or button. The widget displays the bond type resulted from calculation of the tax amount, thus your name of the widget should be Tax.

If you want to check the source code for the widget you just added in, you can try to click on the Form button in the Menu bar, and select View Code.





```
QWidget *centralwidget;  
QPushButton *pushButton_Calculate;  
QLabel *label_ChargableIncome;  
QLineEdit *lineEdit_Income;  
QLabel *label_Tax;  
  
void setupUi(QMainWindow *MainWindow)  
{  
    if (MainWindow->objectName().isEmpty())  
        MainWindow->setObjectName(QStringLiteral("MainWindow"));  
    MainWindow->resize(800, 600);  
    centralwidget = new QWidget(MainWindow);  
    centralwidget->setObjectName(QStringLiteral("centralwidget"));  
    pushButton_Calculate = new QPushButton(centralwidget);  
    pushButton_Calculate->setObjectName(QStringLiteral("pushButton_Calculate"));  
    pushButton_Calculate->setGeometry(QRect(40, 230, 721, 141));  
    QFont font;  
    font.setPointSize(27);  
    pushButton_Calculate->setFont(font);  
    label_ChargableIncome = new QLabel(centralwidget);  
    label_ChargableIncome->setObjectName(QStringLiteral("label_ChargableIncome"));  
    label_ChargableIncome->setGeometry(QRect(40, 20, 261, 41));  
    QFont font1;  
    font1.setPointSize(22);  
    label_ChargableIncome->setFont(font1);  
    lineEdit_Income = new QLineEdit(centralwidget);  
    lineEdit_Income->setObjectName(QStringLiteral("lineEdit_Income"));  
    lineEdit_Income->setGeometry(QRect(40, 70, 721, 151));  
    QFont font2;  
    font2.setPointSize(30);  
    lineEdit_Income->setFont(font2);  
    lineEdit_Income->setAlignment(Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter);
```

Close

### 5.3.2 Import Statements

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5 import uic
import math
```

```
import pandas as pd
```

After finishing the UI design and build, now you can start to link the code for the application with the UI you just created. To start, you need to import all the necessary modules. In Python, importing statements enables you to use the data or functions from a different module.

To do this, you will import the following modules.

The sys model provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter (System-specific parameters and functions, n.d.)

QWidget class provides the basic capability to render the screen. Besides displaying the UI of the function, this module handles the input events(qwidget reference)

PyQt5 module includes the uic that generates the C++ code which creates the user interface (Qt designer reference)

As introduced in the previous section, Pandas contributes to retrieving the tax rate information by calling the HTML retrieve function.

If you want to use the module function by importing the module directly, you need to call the function of the module by using module name followed by function name ("from Tax import \*"). In our case, to call the tax\_to\_pay function in the Tax module,

given that we have a tax class in the Tax.py file, we need to initialize a new instance your\_tax first, and call the function using tax.tax\_to\_pay.

However, you can also choose to simply import the tax function instead of the whole module by typing From Tax import tax, which would only import the tax function and we can implement it directly.

### 5.3.3 Link GUI to Python

```
qtCreatorFile = "IncomeTaxCalculator.ui"  
Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
```

After you import all the modules to be used, you need to link the GUI and the python file to set up the user interface from IncomeTaxCalculator.ui to Ui\_MainWindow. Therefore, we need to store the IncomeTaxCalculator.ui and Tax\_calculation.py in the same folder.

After setting up the user interface from IncomeTaxCalculator.ui to Ui\_MainWindow, the function Tax\_Calculator mentioned above is triggered by the button 'Calculate' to process the calculation of tax to be paid.

There are a few built-in functions used to link the GUI with Python. The functions are shown below:

Text(): This property holds the LineEdit's text. Setting this property clears the selection, clears the undo/redo history, moves the cursor to the end of the line and resets the modified property to false. The text is not validated when inserted with The text is truncated to maxLength() length. By default, this property contains an empty string.

```
income = float(self.lineEdit_Income.text())
```

Clicked.connect(): trigger the tax calculation when click on the button

```
self.pushButton_Calculate.clicked.connect(self.Tax_Calculator)
```

.setText: sets the text of a text field specified by widgetID to text

```
self.label_ChargableIncome.setText(  
    "<span style='font-size:12pt;'"  
    "Please enter your income: "  
    "</span>"  
)
```

```
self.label_Tax.setText(str(tax))
```

## 6. Appendix

### 6.1 Code for Basic User

```
#For Basic User  
  
import math  
gross_tax_payable=[0, 200, 550, 3350, 7950, 13950, 21150,  
28750, 36550, 44550]  
income_tax_rate=[2.0, 3.5, 7.0, 11.5, 15.0, 18.0, 19.0, 19.5,  
20.0, 22.0]  
chargeable_income=[20000, 30000, 40000, 80000, 120000, 160000,  
200000, 240000, 280000, 320000]  
  
income = float(input("Please enter your monthly income: "))  
  
# Append math.inf to chargeable_income.  
chargeable_income.append(math.inf)
```

```

# Find in chargeable_income, the location (i) of the first
number which is greater than x,
i=next(filter(lambda s: s[1]>income,
enumerate(chargeable_income)))[0]
# If i equals 0, tax=0
if i==0:
    tax=0.0
# If i is greater than 0), use gross_tax_payable[i-1],
income_tax_rate[i-1] and chargeable_income[i-1]
# to compute
y=gross_tax_payable[i-1]+income_tax_rate[i-1]/100*(x-chargeabl
e_income[i-1])
else:

tax=gross_tax_payable[i-1]+income_tax_rate[i-1]/100*(income-ch
argeable_income[i-1])

print(f'You should pay ${tax} as your income tax')

```

```

Please enter your monthly income: 25000
You should pay $100.0 as your income tax

```

## 6.2 Code for Premium User

### 6.2.1 Tax.py

```

'''
This is for Tax
'''
class Tax:

```

```

    def __init__(self, your_income, gross_tax_payable,
income_tax_rate, chargeable_income):
        self.gross_tax_payable = gross_tax_payable
        self.income_tax_rate = income_tax_rate
        self.chargeable_income = chargeable_income
        self.your_income = your_income

    def tax_to_pay(self):
        import math
        self.chargeable_income.append(math.inf)
        # Find in xi, the location (i) of the first number
which is greater than x,
        i=next(filter(lambda s: s[1]>self.your_income,
enumerate(self.chargeable_income)))[0]
        # If i equals 0, tax=0
        if i==0:
            tax=0
            # If i is greater than 0), use gross_tax_payable[i-1],
income_tax_rate[i-1] and chargeable_income[i-1]
            # to compute
y=gross_tax_payable[i-1]+income_tax_rate[i-1]/100*(x-chargeabl
e_income[i-1])

        else:

tax=self.gross_tax_payable[i-1]+self.income_tax_rate[i-1]/100*
(self.your_income-self.chargeable_income[i-1])

        return tax

```

### 6.2.2 Tax calculation.py

```

import sys

from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5 import uic

import math

from Tax import *

```



```

qtCreatorFile = "IncomeTaxCalculator.ui"
Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from unicodedata import normalize

tables =
pd.read_html('https://www.iras.gov.sg/irashome/Individuals/Loc
als/Working-Out-Your-Taxes/Income-Tax-Rates/')

IncomeTaxRate_Table=tables[0]

income_tax_rate=[float(x.split()[1]) for x in
IncomeTaxRate_Table['Income Tax Rate (%)']]
chargeable_income=[int(x.split()[1].replace('$','').replace(',
','')) for x in IncomeTaxRate_Table['Chargeable Income'] ]
gross_tax_payable=[int(x.split()[0].replace(',','')) for x in
IncomeTaxRate_Table['Gross Tax Payable ($)']]

class Main(QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.label_ChargableIncome.setText(
            "<span style='font-size:12pt;'"
            "Please enter your income: "
            "</span>"
        )

        self.label_Tax.setText(
            "<span style='font-size:20pt;text-align: center'"

```

```

        "Taxes you should Pay "
        "</span>"
    )

self.pushButton_Calculate.clicked.connect(self.Tax_Calculator)

def Tax_Calculator(self):
    income = float(self.lineEdit_Income.text())

    # Calculate Tax
    your_tax = Tax(income, gross_tax_payable,
income_tax_rate,chargeable_income)
    tax = your_tax.tax_to_pay()
    self.label_Tax.setText(str(tax))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    main = Main()
    main.show()
    sys.exit(app.exec_())

```

### 6.2.3 IncomeTaxCalculator.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>800</width>

```

```
<height>600</height>
</rect>
</property>
<property name="windowTitle">
  <string>Income Tax Calculator</string>
</property>
<widget class="QWidget" name="centralwidget">
  <widget class="QPushButton" name="pushButton_Calculate">
    <property name="geometry">
      <rect>
        <x>40</x>
        <y>230</y>
        <width>721</width>
        <height>141</height>
      </rect>
    </property>
    <property name="font">
      <font>
        <pointsize>27</pointsize>
      </font>
    </property>
    <property name="text">
      <string> Calculate your tax </string>
    </property>
  </widget>
  <widget class="QLabel" name="label_ChargableIncome">
    <property name="geometry">
      <rect>
        <x>40</x>
        <y>20</y>
        <width>261</width>
        <height>41</height>
      </rect>
    </property>
```

```
<property name="font">
  <font>
    <pointsize>22</pointsize>
  </font>
</property>
<property name="text">
  <string> Your Income:</string>
</property>
</widget>
<widget class="QLineEdit" name="lineEdit_Income">
  <property name="geometry">
    <rect>
      <x>40</x>
      <y>70</y>
      <width>721</width>
      <height>151</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>30</pointsize>
    </font>
  </property>
  <property name="text">
    <string>1_000</string>
  </property>
  <property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
</widget>
<widget class="QLabel" name="label_Tax">
  <property name="geometry">
    <rect>
```

```
<x>40</x>
<y>380</y>
<width>721</width>
<height>161</height>
</rect>
</property>
<property name="font">
  <font>
    <pointsize>68</pointsize>
  </font>
</property>
<property name="styleSheet">
  <string notr="true">background-color: rgb(255, 255,
255);</string>
</property>
<property name="text">
  <string/>
</property>
<property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
</widget>
</widget>
</widget>
<resources/>
<connections/>
</ui>
```

## 7. References

GUI.SetText. (n.d.). Retrieved November 14, 2020, from

[http://compsci.ca/holtsoft/doc/gui\\_settext.html](http://compsci.ca/holtsoft/doc/gui_settext.html)

Income Tax Rates. (2020, February 17). Retrieved November 14, 2020, from

<https://www.iras.gov.sg/irashome/Individuals/Locals/Working-Out-Your-Taxes/Income-Tax-Rates/>

Pandas. (n.d.). Retrieved November 14, 2020, from <https://pandas.pydata.org/>

Python 3.9.0 documentation. (2020, November 13). Retrieved November 14, 2020,

from <https://docs.python.org/3/>

Sys - System-specific parameters and functions¶. (n.d.). Retrieved November 14,

2020, from <https://docs.python.org/3/library/sys.html>

What is Python? Executive Summary. (n.d.). Retrieved November 14, 2020, from

<https://www.python.org/doc/essays/blurb/>