# Module 5

**Web Scrapping, Working with Excel Spreadsheets, Working with PDF and Word Documents, Working with CSV files and JSON data**

# What is Web scraping

- Web Scraping is an automatic method to obtain large amounts of data from websites.
- Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications
- Web scraping is the term for using a program to download and process content from the Web.
- **For example**, Google runs many web scraping programs to index web pages for its search engine

# Crawler and Scrapper

- Web scraping requires two parts namely the **crawler** and the **scraper**.
- The **crawler** is an artificial intelligence algorithm that browses the web to search the particular data required by following the links across the internet.
- The **scraper,** on the other hand, is a specific tool created to extract the data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data.

# Uses of Web Scraping

1. Price Monitoring
2. Market Research
3. News Monitoring
4. Sentiment Analysis
5. **Email Marketing**

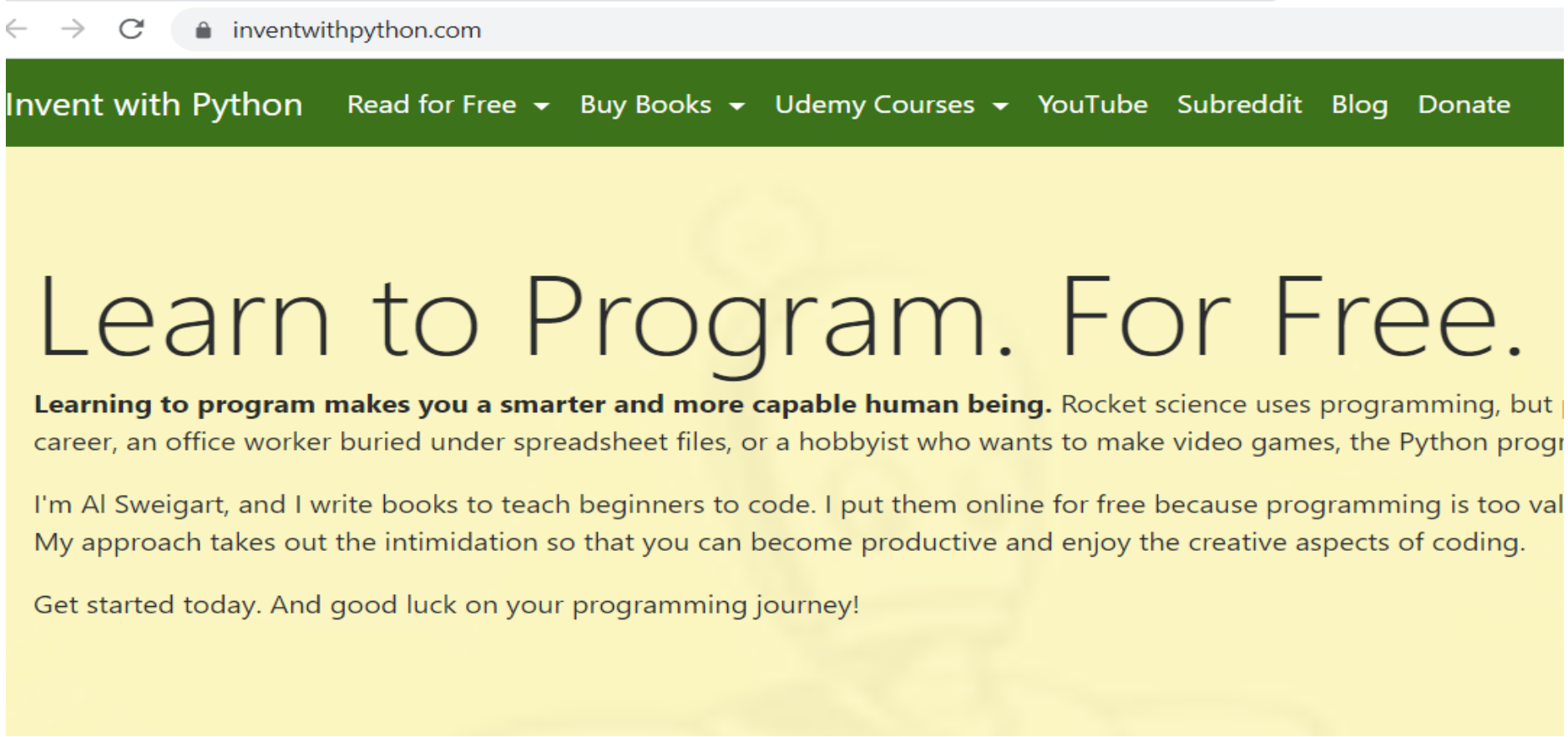
# Modules used for Webscraping

1. **webbrowser**: Comes with Python and opens a browser to a specific page.
2. **requests**: Downloads files and web pages from the Internet.
3. **beautiful Soup**: Parses HTML, the format that web pages are written in.
4. **selenium**: Launches and controls a web browser. Selenium is able to fill in forms and simulate mouse clicks in this browser.

# Project: **maplt.py with the webbrowser Module**

```
import webbrowser
webbrowser.open('http://inventwithpython.com/')
```

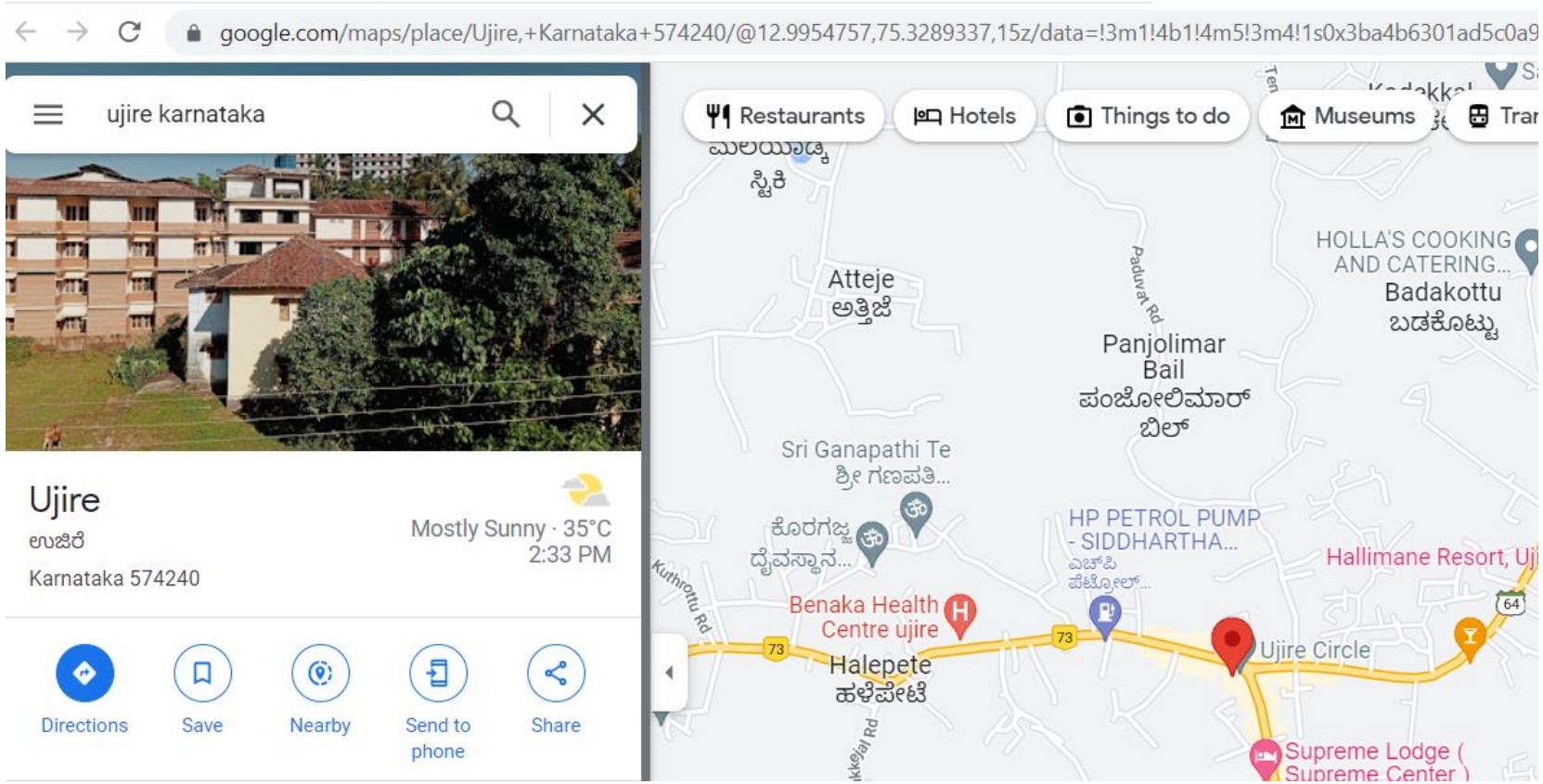
```
True
```



```python
import webbrowser
addressline = ['ujire','karnataka']
address = ' '.join(addressline[:])
webbrowser.open('https://www.google.com/maps/place/'+address)
```

```
True
```

Example :



# DOWNLOADING FILES FROM THE WEB WITH THE REQUESTS MODULE

```
import requests
res = requests.get('https://automatetheboringstuff.com/files/rj.txt')
```

```
len(res.text)
```

```
178978
```

```
print(res.text[:250])
```

```
The Project Gutenberg EBook of Romeo and Juliet, by William Shakespeare

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Projec
```

# Saving Downloaded Files to the Hard Drive

1. **Call requests.get()** to download the file.
2. **Call open()** with '**wb**' to create a new file in write binary mode.
3. Loop over the Response object's **iter_content() method**.
4. **Call write()** on each iteration to write the content to the file.
5. **Call close()** to close the file.

# Example:

```
import requests
res = requests.get('https://automatetheboringstuff.com/files/rj.txt')
res.raise_for_status() # to ensure that a program halts if a bad download occurs
playFile = open('RomeoAndJuliet.txt', 'wb')
for chunk in res.iter_content(100000):
    playFile.write(chunk)
playFile.close()
```

# PARSING HTML WITH THE BEAUTIFULSOUP MODULE

*example - Notepad

File Edit Format View Help

```
<!-- This is the example.html file. -->

<html>
  <head>
        <title>The Website Title</title>
  </head>

  <body>
        <p>Download my <strong>Python</strong> book from
        <a href="http://inventwithpython.com">my website</a>.</p>

        <p class="slogan">Learn Python the easy way!</p>

        <p>By <span id="author">Al Sweigart</span></p>

  </body>
</html>
```

File | C:/Users/thyagu/18CS55/example.html

Download my **Python** book from my website.

Learn Python the easy way!

By Al Sweigart

# PARSING HTML WITH THE BEAUTIFULSOUP MODULE

```
import bs4
exampleFile = open('example.html')
exampleSoup = bs4.BeautifulSoup(exampleFile)
type(exampleSoup)
```

```
bs4.BeautifulSoup
```

# FINDING AN ELEMENT WITH THE SELECT() METHOD

- **soup.select('div’)** : All elements named 
- **soup.select('#author’)** : The element with an **id** attribute of **author**
- **soup.select('.notice’):** All elements that use a CSS **class** attribute named **notice**
- **soup.select('div span’):** All elements named **** that are within an element named ** **
- **soup.select('input[type="button"]’):** All elements named **<input>**that have an attribute named **type** with value **button**

```python
import bs4
exampleFile = open('example.html')
exampleSoup = bs4.BeautifulSoup(exampleFile.read())
elems = exampleSoup.select('#author')
print(elems)
```

```
[<span id="author">Al Sweigart</span>]
```

```python
elems[0].getText()
```

```
'Al Sweigart'
```

```python
elems[0].attrs
```

```
{'id': 'author'}
```

```python
pElems = exampleSoup.select('p')
str(pElems[0])
```

```
'<p>Download my <strong>Python</strong> book from <a href="http://inventwithpython.com">my website</a>.</p>'
```

```python
str(pElems[1])
```

```
'<p class="slogan">Learn Python the easy way!</p>'
```

```python
str(pElems[2])
```

```
'<p>By <span id="author">Al Sweigart</span></p>'
```

```
pElems[0].getText()
```

```
'Download my Python book from my website.'
```

```
pElems[1].getText()
```

```
'Learn Python the easy way!'
```

```
pElems[2].getText()
```

```
'By Al Sweigart'
```

# GETTING DATA FROM AN ELEMENT'S ATTRIBUTES

```
import bs4
soup = bs4.BeautifulSoup(open('example.html'))
spanElem = soup.select('span')[0]
str(spanElem)
```

```
'<span id="author">Al Sweigart</span>'
```

```
spanElem.get('id')
```

```
'author'
```

```
spanElem.get('some_nonexistent_addr') == None
```

```
True
```

```
spanElem.attrs
```

```
{'id': 'author'}
```

# PROJECT: GOOGLE SEARCH

```
import requests
import bs4
text= input("Enter searching string: ")
url = 'https://google.com/search?q=' + text
request_result=requests.get( url )
```

```
Enter searching string: Python
```

# PROJECT: GOOGLE SEARCH

```
soup = bs4.BeautifulSoup( request_result.text, "html.parser" )
print(soup)
```

```
<!DOCTYPE html>
<html lang="en-IN"><head><meta charset="utf-8"/><meta content="/images/branding/googleg/1x/googleg_s
tandard_color_128dp.png" itemprop="image"/><title>Python - Google Search</title><script nonce="gjaXj
qoJTqCfRnpnnzfcoQ">(function(){
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAttri
bute("data-submitfalse");a="1"===c||"q"===c&&!a.elements.q.value?!0:!1}else a=!1;a&&(b.preventDefaul
t(),b.stopPropagation())},!0);document.documentElement.addEventListener("click",function(b){var a;a:
{for(a=b.target;a&&a!==document.documentElement;a=a.parentElement)if("A"===a.tagName){a="1"===a.getA
ttribute("data-nohref");break a}a=!1}a&&b.preventDefault()},!0);}).call(this);(function(){var a=wind
```

# PROJECT: GOOGLE SEARCH

```
# soup.find.all( h3 ) to grab
# all major headings of our search result,
heading_object=soup.find_all('h3' )

# Iterate through the object
# and print it as a string.
for info in heading_object:
    print(info.getText())
    print("------")
```

# PROJECT: GOOGLE SEARCH

```
Python.org
------
Python Tutorial - W3Schools
------
Python (programming language) - Wikipedia
------
Online Python Compiler (Interpreter) - Programiz
------
Python
------
CPython
------
Python Tutorial - Tutorialspoint
------
Python Courses & Tutorials - Codecademy
------
Learn Python - Free Interactive Python Tutorial
------
Learn Python Tutorials - Kaggle
------
```

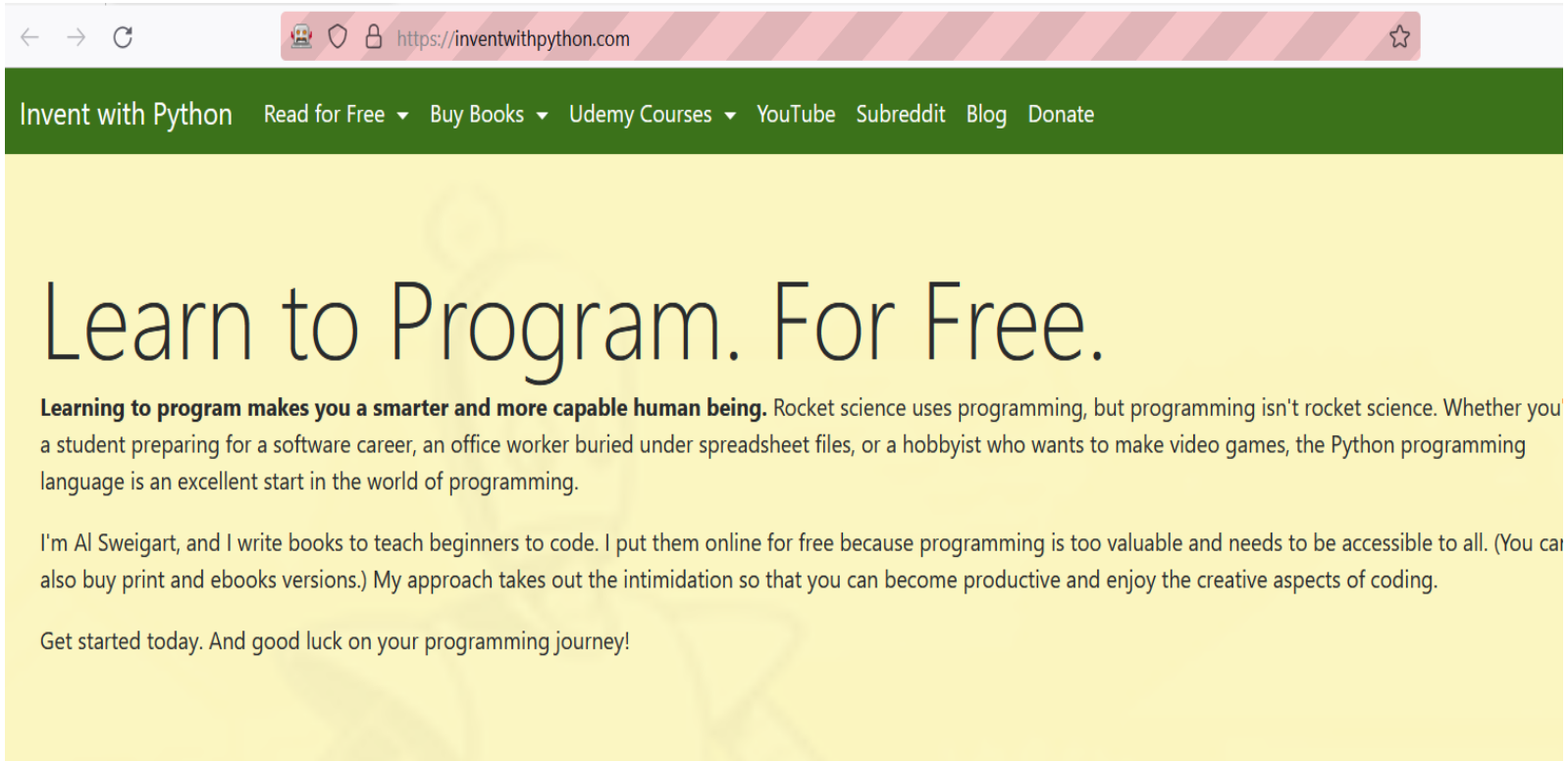
# STARTING A SELENIUM-CONTROLLED BROWSER

```
pip install selenium
```

```
Collecting selenium
  Downloading selenium-4.7.2-py3-none-any.whl (6.3 MB)
     ---------------------------------------- 6.3/6.3 MB 600.5 kB/s eta 0:00:00
Collecting trio-websocket~=0.9
  Downloading trio_websocket-0.9.2-py3-none-any.whl (16 kB)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\thyagu\anaconda3\li
m selenium) (2022.9.14)
Collecting trio~=0.17
  Downloading trio-0.22.0-py3-none-any.whl (384 kB)
     ---------------------------------------- 384.9/384.9 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: urllib3[socks]~=1.26 in c:\users\thyagu\anaconda3\
```

# STARTING A SELENIUM-CONTROLLED BROWSER

```
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
```



# FINDING ELEMENTS ON THE PAGE

| Method name | WebElement object/list returned |
| --- | --- |
| **browser.find_element_by_class_name(*name*)**<br>**browser.find_elements_by_class_name(*name*)** | Elements that use the CSS class *name* |
| **browser.find_element_by_css_selector(*selector*)**<br>**browser.find_elements_by_css_selector(*selector*)** | Elements that match the CSS *selector* |
| **browser.find_element_by_id(*id*)**<br>**browser.find_elements_by_id(*id*)** | Elements with a matching *id* attribute value |
| **browser.find_element_by_link_text(*text*)**<br>**browser.find_elements_by_link_text(*text*)** | <a> elements that completely match the *text* provided |
| **browser.find_element_by_partial_link_text(*text*)**<br>**browser.find_elements_by_partial_link_text(*text*)** | <a> elements that contain the *text* provided |
| **browser.find_element_by_name(*name*)**<br>**browser.find_elements_by_name(*name*)** | Elements with a matching *name* attribute value |
| **browser.find_element_by_tag_name(*name*)**<br>**browser.find_elements_by_tag_name(*name*)** | Elements with a matching tag *name* (case insensitive; an <a> element is matched by 'a' and 'A') |

| Attribute or method | Description |
| --- | --- |
| **tag_name** | The tag name, such as 'a' for an <a> element |
| **get_attribute(*name*)** | The value for the element’s name attribute |
| **text** | The text within the element, such as 'hello' in hello |
| **clear()** | For text field or text area elements, clears the text typed into it |
| **is_displayed()** | Returns True if the element is visible; otherwise returns False |
| **is_enabled()** | For input elements, returns True if the element is enabled; otherwise returns False |
| **is_selected()** | For checkbox or radio button elements, returns True if the element is selected; otherwise returns False |
| **location** | A dictionary with keys 'x' and 'y' for the position of the element in the page |

# Example Selenium

```python
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('bookcover')
    print('Found <%s> element with that class name!' % (elem.tag_name))
except:
    print('Was not able to find an element with that name.')
```

```
Found <img> element with that class name!
```

```python
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('bookcover')
    print('Found <%s> element with that class name!' % (elem.tag_name))
except:
    print('Was not able to find an element with that name.')
```

```
Was not able to find an element with that name.
```

# Clicking the Page

- WebElement objects returned from the find_element_* and find_elements_* methods have a **click() method** that simulates a mouse click on that element.
- This method can be used to follow a link, **make a selection on a radio button, click a Submit button**, or trigger whatever else might happen when the element is clicked by the mouse.

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://inventwithpython.com')
>>> linkElem = browser.find_element_by_link_text('Read It Online')
>>> type(linkElem)
<class 'selenium.webdriver.remote.webelement.WebElement'>
>>> linkElem.click() # follows the "Read It Online" link
```

# Filling Out and Submitting Forms

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('https://mail.yahoo.com')
>>> emailElem = browser.find_element_by_id('login-username')
>>> emailElem.send_keys('not_my_real_email')
>>> passwordElem = browser.find_element_by_id('login-passwd')
>>> passwordElem.send_keys('12345')
>>> passwordElem.submit()
```

# Sending Special Keys

| Attributes | Meanings |
| --- | --- |
| **Keys.DOWN, Keys.UP, Keys.LEFT, Keys.RIGHT** | **The keyboard arrow keys** |
| **Keys.ENTER, Keys.RETURN** | **The ENTER and RETURN keys** |
| **Keys.HOME, Keys.END, Keys.PAGE_DOWN, Keys.PAGE_UP** | **The home, end, pagedown, and pageup keys** |
| **Keys.ESCAPE, Keys.BACK_SPACE, Keys.DELETE** | **The ESC, BACKSPACE, and DELETE keys** |
| **Keys.F1, Keys.F2,..., Keys.F12** | **The F1 to F12 keys at the top of the keyboard** |
| **Keys.TAB** | **The TAB key** |

```
>>> from selenium import webdriver
>>> from selenium.webdriver.common.keys import Keys
>>> browser = webdriver.Firefox()
>>> browser.get('http://nostarch.com')
>>> htmlElem = browser.find_element_by_tag_name('html')
>>> htmlElem.send_keys(Keys.END)     # scrolls to bottom
>>> htmlElem.send_keys(Keys.HOME)    # scrolls to top
```

# Clicking Browser Buttons

Selenium can simulate clicks on various browser buttons as well through the following methods:

- `browser.back()`. Clicks the Back button.
- `browser.forward()`. Clicks the Forward button.
- `browser.refresh()`. Clicks the Refresh/Reload button.
- `browser.quit()`. Clicks the Close Window button.

# More Information on Selenium

Selenium can do much more beyond the functions described here. It can modify your browser's cookies, take screenshots of web pages, and run custom JavaScript. To learn more about these features, you can visit the Selenium documentation at *http://selenium-python.readthedocs.org/*.

# 5.2. Working with Excel Spreadsheets

example.xlsx

| | A | B | C |
|---|---|---|---|
| 1 | 4/5/2015 13:34 | Apples | 73 |
| 2 | 4/5/2015 3:41 | Cherries | 85 |
| 3 | 4/6/2015 12:46 | Pears | 14 |
| 4 | 4/8/2015 8:59 | Oranges | 52 |
| 5 | 4/10/2015 2:07 | Apples | 152 |
| 6 | 4/10/2015 18:10 | Bananas | 23 |
| 7 | 4/10/2015 2:40 | Strawberri | 98 |
| 8 | | | |

Sheet1 Sheet2 Sheet3

# 5.2. Working with Excel Spreadsheets

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
wb.get_sheet_names()
```

```
['Sheet1', 'Sheet2', 'Sheet3']
```

# GETTING CELLS FROM THE SHEETS

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
sheet = wb.get_sheet_by_name('Sheet1')
```

```
sheet['A1'].value
```

```
datetime.datetime(2015, 4, 5, 13, 34, 2)
```

```
sheet['B1'].value
```

```
'Apples'
```

```
sheet['C1'].value
```

```
73
```

# Iterate through Excel rows

```python
# import module
import openpyxl
# load excel with its path
wrkbk = openpyxl.load_workbook("example.xlsx")
sh = wrkbk.active
# iterate through excel and display data
for i in range(1, sh.max_row+1):
    print("\n")
    print("Row ", i, " data :")
    for j in range(1, sh.max_column+1):
        cell_obj = sh.cell(row=i, column=j)
        print(cell_obj.value, end=" ")
```

```
Row  1  data :
2015-04-05 13:34:02 Apples 73

Row  2  data :
2015-04-05 03:41:23 Cherries 85

Row  3  data :
2015-04-06 12:46:51 Pears 14

Row  4  data :
2015-04-08 08:59:43 Oranges 52

Row  5  data :
2015-04-10 02:07:00 Apples 152

Row  6  data :
2015-04-10 18:10:37 Bananas 23

Row  7  data :
2015-04-10 02:40:46 Strawberries 98
```

# CREATING AND SAVING EXCEL DOCUMENTS

```
import openpyxl
wb = openpyxl.Workbook()
wb.get_sheet_names()
```

```
['Sheet']
```

```
sheet = wb.active
sheet.title
```

```
'Sheet'
```

```
sheet.title = 'Spam Bacon Eggs Sheet'
wb.get_sheet_names()
```

```
['Spam Bacon Eggs Sheet']
```

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
sheet = wb.active
sheet.title = 'Spam Spam Spam'
wb.save('example_copy.xlsx')
```

| | A | B | C |
|---|---|---|---|
| 1 | 4/5/2015 13:34 | Apples | 73 |
| 2 | 4/5/2015 3:41 | Cherries | 85 |
| 3 | 4/6/2015 12:46 | Pears | 14 |
| 4 | 4/8/2015 8:59 | Oranges | 52 |
| 5 | 4/10/2015 2:07 | Apples | 152 |
| 6 | 4/10/2015 18:10 | Bananas | 23 |
| 7 | 4/10/2015 2:40 | Strawberri | 98 |
| 8 | | | |
| 9 | | | |

Spam Spam Spam | Sheet2 | Sheet3

# CREATING AND REMOVING SHEETS

```
import openpyxl
wb = openpyxl.Workbook()
wb.get_sheet_names()
```

```
['Sheet']
```

```
wb.create_sheet()
```

```
<Worksheet "Sheet1">
```

```
wb.get_sheet_names()
```

```
['Sheet', 'Sheet1']
```

```
wb.create_sheet(index=0, title='First Sheet')
```

```
<Worksheet "First Sheet">
```

```
wb.get_sheet_names()
```

```
['First Sheet', 'Sheet', 'Sheet1']
```

```
wb.create_sheet(index=2, title='Middle Sheet')
```

```
<Worksheet "Middle Sheet">
```

```
wb.get_sheet_names()
```

```
['First Sheet', 'Sheet', 'Middle Sheet', 'Sheet1']
```

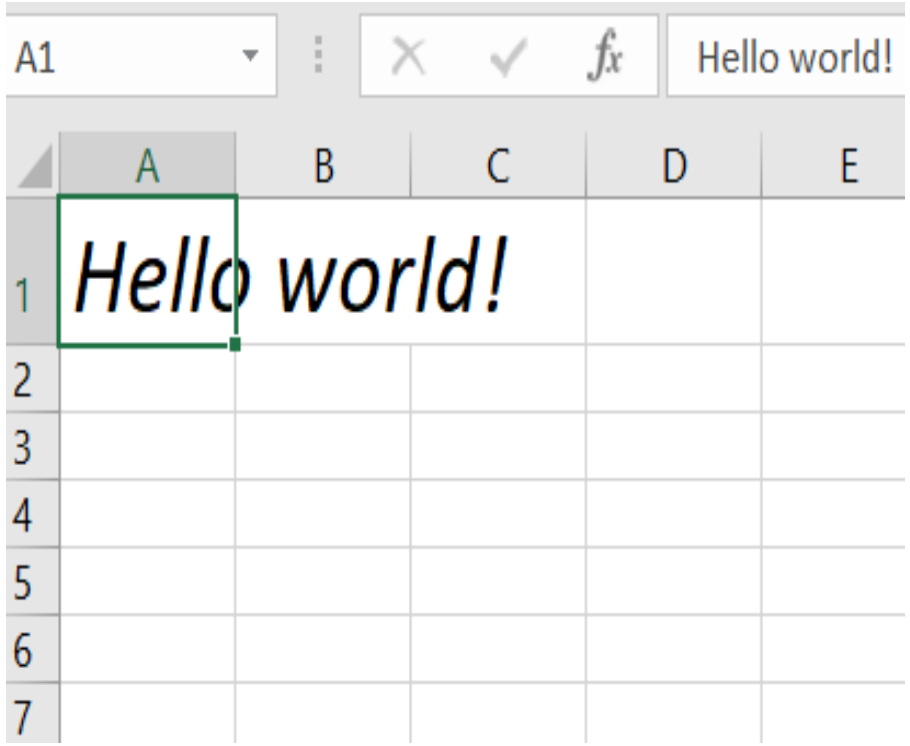
# WRITING VALUES TO CELLS

```
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.get_sheet_by_name('Sheet')
sheet['A1'] = 'Hello world!'
sheet['A1'].value
```

```
'Hello world!'
```

# SETTING THE FONT STYLE OF CELLS

```python
import openpyxl
from openpyxl.styles import Font
wb = openpyxl.Workbook()
sheet = wb.get_sheet_by_name('Sheet')
italic24Font = Font(size=24, italic=True)
sheet['A1'].font = italic24Font
sheet['A1'] = 'Hello world!'
wb.save('styled.xlsx')
```



# Font Objects

To set font style attributes, you pass keyword arguments to **Font().** Table below shows the possible keyword arguments for the **Font()** function

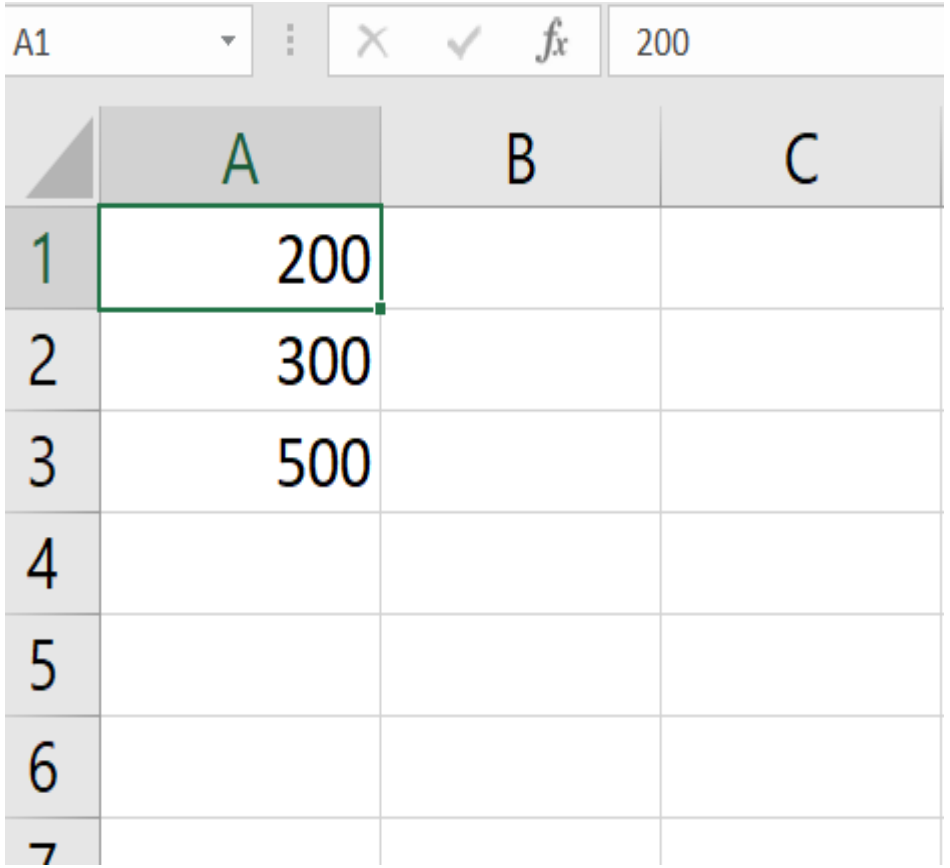
| Keyword argument | Data type | Description |
|---|---|---|
| name | String | The font name, such as 'Calibri' or 'Times New Roman' |
| size | Integer | The point size |
| bold | Boolean | True, for bold font |
| italic | Boolean | True, for italic font |

```python
import openpyxl
from openpyxl.styles import Font
wb = openpyxl.Workbook()
sheet = wb.get_sheet_by_name('Sheet')
fontObj1 = Font(name='Times New Roman', bold=True)
sheet['A1'].font = fontObj1
sheet['A1'] = 'Bold Times New Roman'
fontObj2 = Font(size=24, italic=True)
sheet['B3'].font = fontObj2
sheet['B3'] = '24 pt Italic'
wb.save('styles.xlsx')
```

# FORMULAS

**Formulas, which begin with an equal sign, can configure cells to contain values calculated from other cells.**

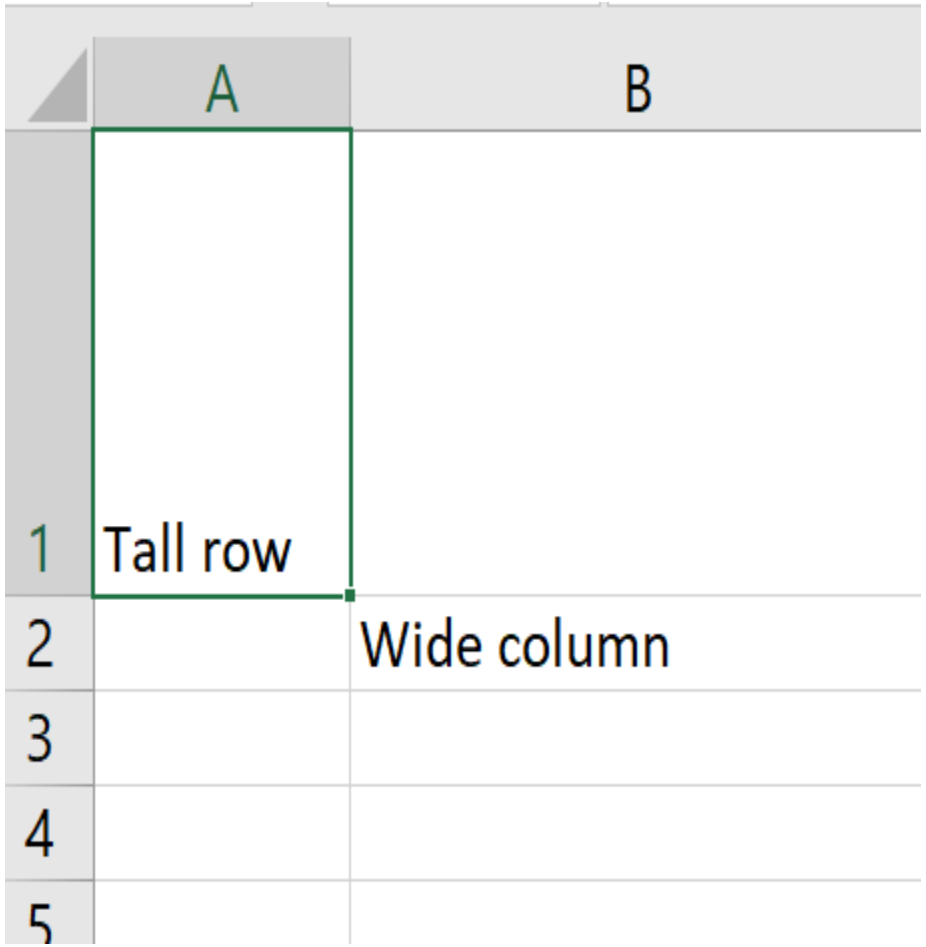
```
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.active
sheet['A1'] = 200
sheet['A2'] = 300
sheet['A3'] = '=SUM(A1:A2)'
wb.save('writeFormula.xlsx')
```



# Adjusting Rows and Columns

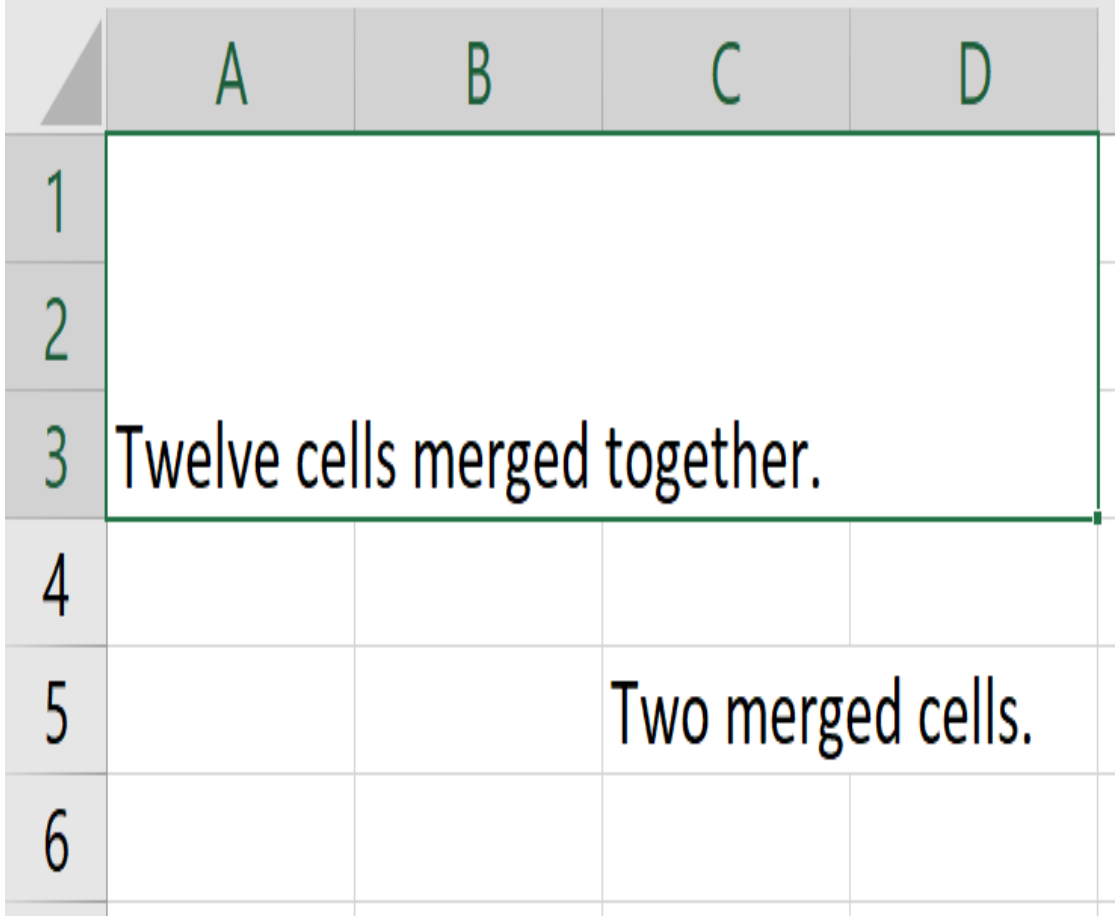
## Setting Row Height and Column Width

```python
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.active
sheet['A1'] = 'Tall row'
sheet['B2'] = 'Wide column'
sheet.row_dimensions[1].height = 70
sheet.column_dimensions['B'].width = 20
wb.save('dimensions.xlsx')
```



# Merging and Unmerging Cells

```python
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.active
sheet.merge_cells('A1:D3')
sheet['A1'] = 'Twelve cells merged together.
sheet.merge_cells('C5:D5')
sheet['C5'] = 'Two merged cells.'
wb.save('merged.xlsx')
```



# Merging and Unmerging Cells

```python
import openpyxl
wb = openpyxl.load_workbook('merged.xlsx')
sheet = wb.active
sheet.unmerge_cells('A1:D3')
sheet.unmerge_cells('C5:D5')
wb.save('merged.xlsx')
```

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Twelve cells merged together. | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | Two merged cells. | |
| 6 | | | | |
| 7 | | | | |

# Freeze Panes

**For spreadsheets too large to be displayed all at once, it's helpful to "freeze" a few of the top rows or leftmost columns onscreen. Frozen column or row headers, for example, are always visible to the user even as they scroll through the spreadsheet. These are known as *freeze panes*.**

| freeze_panes setting | Rows and columns frozen |
|---|---|
| sheet.freeze_panes = 'A2' | Row 1 |
| sheet.freeze_panes = 'B1' | Column A |
| sheet.freeze_panes = 'C1' | Columns A and B |
| sheet.freeze_panes = 'C2' | Row 1 and columns A and B |
| sheet.freeze_panes = 'A1' or sheet.freeze_panes = None | No frozen panes |

# Freeze Panes

| freeze_panes setting | Rows and columns frozen |
| --- | --- |
| sheet.freeze_panes = 'A2' | Row 1 |
| sheet.freeze_panes = 'B1' | Column A |
| sheet.freeze_panes = 'C1' | Columns A and B |
| sheet.freeze_panes = 'C2' | Row 1 and columns A and B |
| sheet.freeze_panes = 'A1' or sheet.freeze_panes = None | No frozen panes |

| | A | B | C | D |
|---|---|---|---|---|
| 1 | PRODUCE | COST PER POUND | POUNDS SOLD | TOTAL |
| 2 | Potatoes | 0.86 | 21.6 | 18.58 |
| 3 | Okra | 2.26 | 38.6 | 87.24 |
| 4 | Fava beans | 2.69 | 32.8 | 88.23 |
| 5 | Watermelon | 0.66 | 27.3 | 18.02 |
| 6 | Garlic | 1.19 | 4.9 | 5.83 |
| 7 | Parsnips | 2.27 | 1.1 | 2.5 |
| 8 | Asparagus | 2.49 | 37.9 | 94.37 |
| 9 | Avocados | 3.23 | 9.2 | 29.72 |
| 10 | Celery | 3.07 | 28.9 | 88.72 |
| 11 | Okra | 2.26 | 40 | 90.4 |
| 12 | Spinach | 4.12 | 30 | 123.6 |
| 13 | Cucumber | 1.07 | 36 | 38.52 |
| 14 | Apricots | 3.71 | 29.4 | 109.07 |
| 15 | Okra | 2.26 | 9.5 | 21.47 |
| 16 | Fava beans | 2.69 | 5.3 | 14.26 |

# Freeze Panes

```
import openpyxl
wb = openpyxl.load_workbook('produceSales.xlsx')
sheet = wb.active
sheet.freeze_panes = 'A2'
wb.save('freezeExample.xlsx')
```

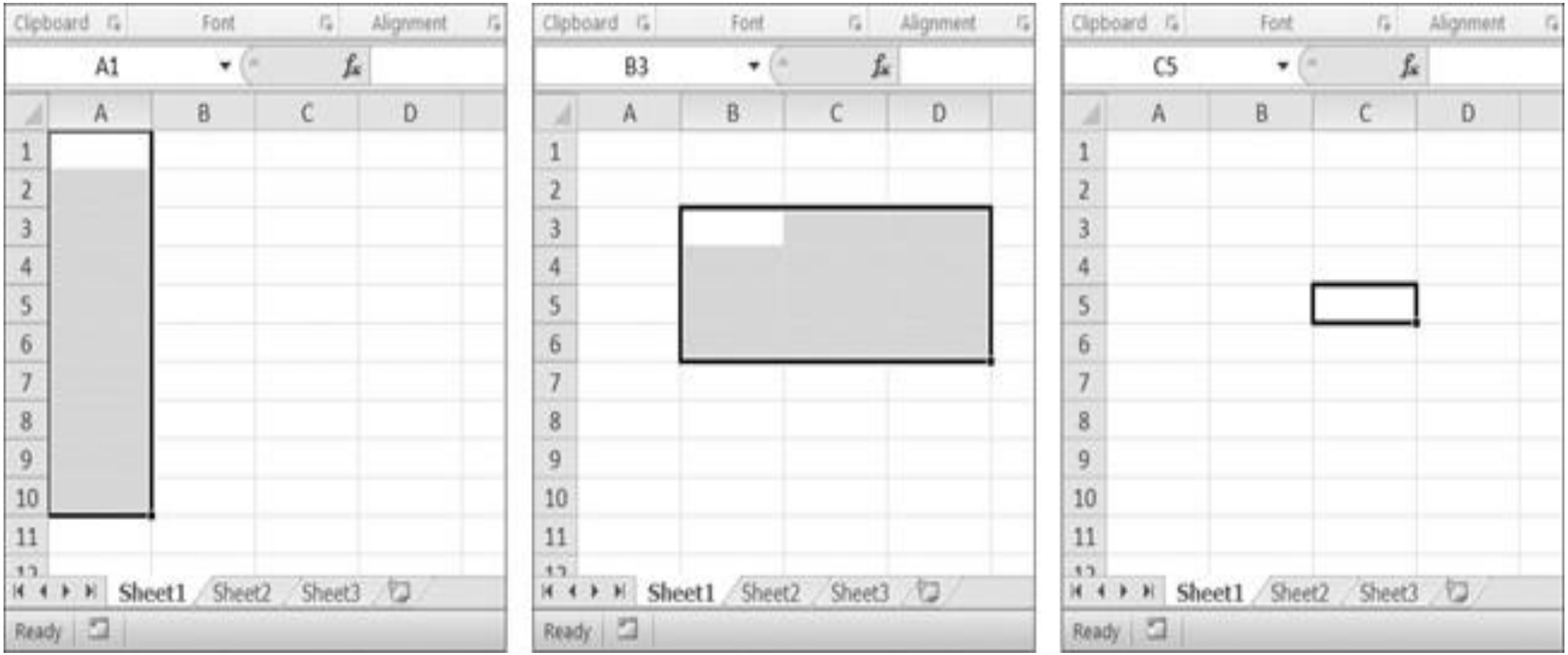
| | A | B | C | D |
|---|---|---|---|---|
| 1 | **PRODUCE** | **COST PER POUND** | **POUNDS SOLD** | **TOTAL** |
| 17 | Watermelon | 0.66 | 35.4 | 23.36 |
| 18 | Ginger | 5.13 | 14.4 | 73.87 |
| 19 | Corn | 1.07 | 12.2 | 13.05 |
| 20 | Grapefruit | 0.76 | 35.7 | 27.13 |
| 21 | Ginger | 5.13 | 15.2 | 77.98 |
| 22 | Eggplant | 2.32 | 5 | 11.6 |
| 23 | Cucumber | 1.07 | 31.8 | 34.03 |
| 24 | Green cabbage | 0.8 | 2.8 | 2.24 |
| 25 | Eggplant | 2.32 | 32.8 | 76.1 |
| 26 | Yellow peppers | 2.87 | 26.5 | 76.06 |
| 27 | Garlic | 1.19 | 38.2 | 45.46 |
| 28 | Grapes | 2.63 | 17.4 | 45.76 |
| 29 | Watermelon | 0.66 | 7.3 | 4.82 |

# CHARTS

OpenPyXL supports creating **bar, line, scatter, and pie charts** using the data in a sheet's cells. To make a chart, you need to do the following:

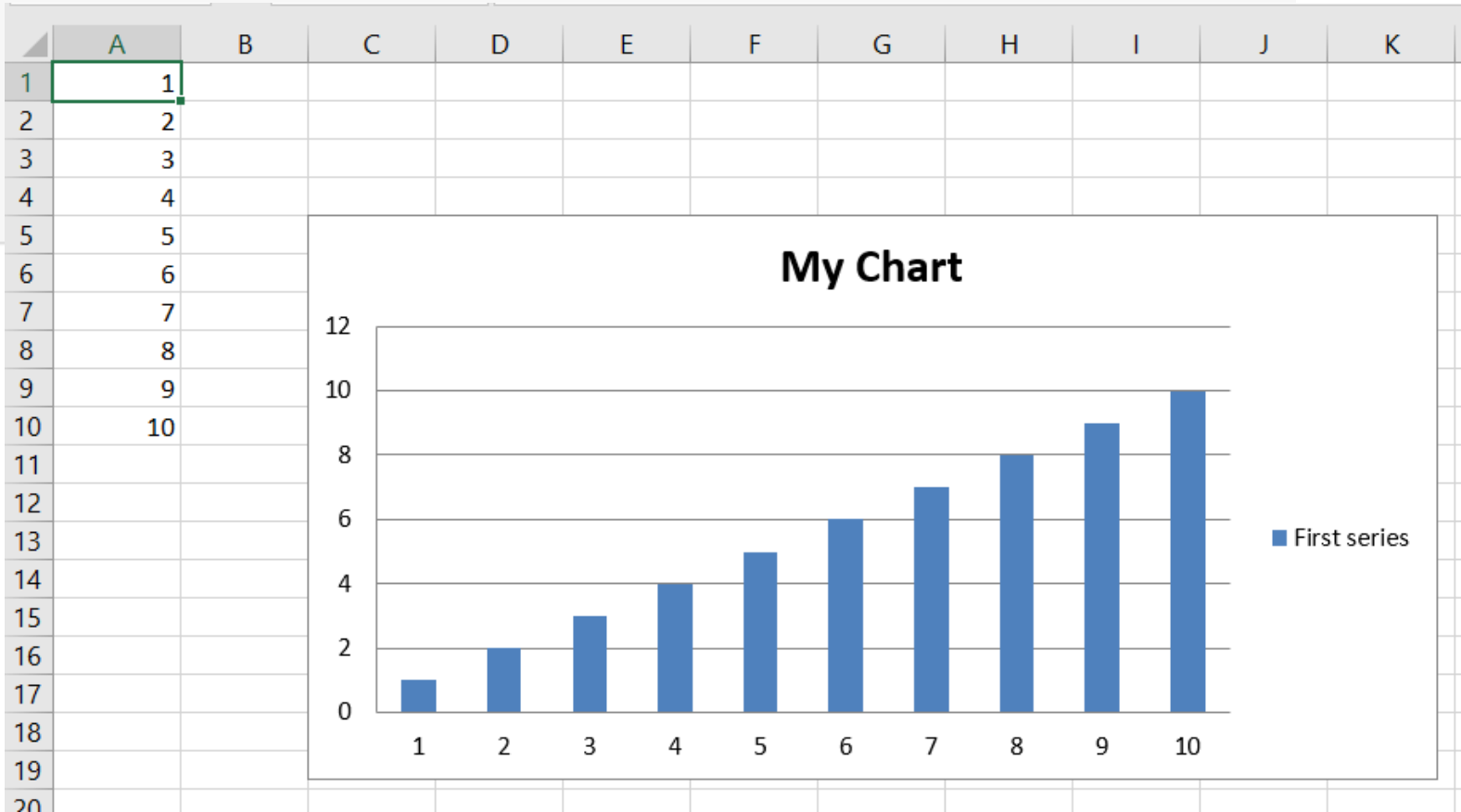
1. Create a **Reference object** from a rectangular selection of cells.
2. Create a **Series object** by passing in the **Reference object**.
3. Create a **Chart object**.
4. Append the **Series object** to the **Chart object**.
5. Add the **Chart object** to the **Worksheet object**, optionally specifying which cell the top left corner of the chart should be positioned..

Figure : From left to right:
**(1, 1), (10, 1);**
**(3, 2), (6, 4);**
**(5, 3), (5, 3)**



```python
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.active
for i in range(1, 11):          # create some data in column A
    sheet['A' + str(i)] = i
refObj = openpyxl.chart.Reference(sheet, min_col=1, min_row=1, max_col=1, max_row=10)
seriesObj = openpyxl.chart.Series(refObj, title='First series')
chartObj = openpyxl.chart.BarChart()
chartObj.title = 'My Chart'
chartObj.append(seriesObj)
sheet.add_chart(chartObj, 'C5')
wb.save('sampleChart.xlsx')
```

```python
import openpyxl
wb = openpyxl.Workbook()
sheet = wb.active
for i in range(1, 11):          # create some data in column A
    sheet['A' + str(i)] = i
refObj = openpyxl.chart.Reference(sheet, min_col=1, min_row=1, max_col=1, max_row=10)
seriesObj = openpyxl.chart.Series(refObj, title='First series')
chartObj = openpyxl.chart.BarChart()
chartObj.title = 'My Chart'
chartObj.append(seriesObj)
sheet.add_chart(chartObj, 'C5')
wb.save('sampleChart.xlsx')
```



# 5.3 Working with PDF and Word Documents

- PDF and Word documents are binary files, which makes them much more complex than plaintext files.
- PDF stands for Portable Document Format and uses the .pdf file extension.

Example

# EXTRACTING TEXT FROM PDFS

```
import PyPDF2
pdfFileObj = open('meetingminutes.pdf', 'rb')
pdf = PyPDF2.PdfReader(pdfFileObj)
len(pdf.pages)
```

```
19
```

```
pageObj = pdf.pages[1]
pageObj.extract_text
```

```
<bound method PageObject.extract_text of {'/Contents': IndirectObject(65, 0, 2445572675088), '/CropBo
x': [0, 0, 612, 792], '/MediaBox': [0, 0, 612, 792], '/Parent': IndirectObject(953, 0, 244557267508
8), '/Resources': {'/ColorSpace': {'/CS0': IndirectObject(975, 0, 2445572675088)}, '/Font': {'/TT0':
IndirectObject(68, 0, 2445572675088), '/TT1': IndirectObject(67, 0, 2445572675088)}}, '/Rotate': 0,
'/StructParents': 1, '/Type': '/Page'}>
```

# DECRYPTING PDFS

encrypted.pdf

```
import PyPDF2
pdfReader = PyPDF2.PdfReader(open('encrypted.pdf', 'rb'))
pdfReader.is_encrypted
```

```
True
```

```
pdfReader.pages[0]
```

```
---------------------------------------------------------------------------
FileNotDecryptedError                     Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9928\137496788.py in <module>
----> 1 pdfReader.pages[0]
```

# DECRYPTING PDFS

```
pdfReader.decrypt('rosebud')
```

```
<PasswordType.OWNER_PASSWORD: 2>
```

```
pdfReader.pages[0]
```

```
{'/CropBox': [0, 0, 612, 792],
 '/Parent': {'/Parent': {'/Type': '/Pages',
   '/Count': 19,
   '/Kids': [IndirectObject(4, 0, 2318172725552),
    IndirectObject(36, 0, 2318172725552),
    IndirectObject(47, 0, 2318172725552)]},
  '/Type': '/Pages',
  '/Count': 9,
  '/Kids': [IndirectObject(72, 0, 2318172725552),
   IndirectObject(3, 0, 2318172725552),
   IndirectObject(17, 0, 2318172725552),
   IndirectObject(20, 0, 2318172725552),
```

# CREATING PDFS

- Open **one or more existing PDFs** (the source PDFs) into **PdfReader** objects.
- Create a new **PdfWriter object**.
- Copy pages from the **PdfReader** objects into the **PdfWriter** object.
- Finally, use the **PdfWriter** object to write the output PDF.

# COPYING PAGES

```
import PyPDF2
pdf1File = open('meetingminutes.pdf','rb')
pdf2File = open('meetingminutes2.pdf','rb')
pdf1Reader = PyPDF2.PdfReader(pdf1File)
pdf2Reader = PyPDF2.PdfReader(pdf2File)
pdfWriter = PyPDF2.PdfWriter()
```

```python
for pageNum in range(len(pdf1Reader.pages)):
    pageObj = pdf1Reader.pages[pageNum]
    pdfWriter.add_page(pageObj)
```

```python
for pageNum in range(len(pdf2Reader.pages)):
    pageObj = pdf2Reader.pages[pageNum]
    pdfWriter.add_page(pageObj)
```

```
pdfOutputFile = open('combinedminutes.pdf', 'wb')
pdfWriter.write(pdfOutputFile)
pdfOutputFile.close()
pdf1File.close()
pdf2File.close()
```

# ROTATING PAGES

```
import PyPDF2
minutesFile = open('meetingminutes.pdf', 'rb')
```

```
pdfReader = PyPDF2.PdfReader(minutesFile)
page = pdfReader.pages[0]
page.rotate(90)
```

```
pdfWriter = PyPDF2.PdfWriter()
pdfWriter.add_page(page)
resultPdfFile = open('rotatedPage.pdf', 'wb')
pdfWriter.write(resultPdfFile)
resultPdfFile.close()
minutesFile.close()
```

# Working WORD DOCUMENTS

*pip install python-docx*

1. Writing to Word documents
2. Reading Word Documents

# Writing WORD DOCUMENTS

```
import docx
doc = docx.Document()
doc.add_heading('Heading for the document', 0)
doc_para = doc.add_paragraph('Your paragraph goes here, ')
doc_para.add_run('hey there, bold here').bold = True
doc_para.add_run(', and ')
doc_para.add_run('these words are italic').italic = True
doc_para.add_run('End of the Paragraph1')
# add a heading of level 2
doc.add_heading('2nd Paragraph', 2)
doc_para = doc.add_paragraph('2nd paragraph starts here, ')
doc_para.add_run(' and ')
doc_para.add_run('these words are regular')
doc_para.add_run('End of the Paragraph 2')
# now save the document to a location
doc.save("one.docx")
```

# Output

# Heading for the document

Your paragraph goes here, **hey there, bold here**, and *these words are italic*End of the Paragraph1

## 2nd Paragraph

2nd paragraph starts here,  and these words are regularEnd of the Paragraph 2

# Reading Word Documents

```python
from docx import Document

doc = Document('one.docx')
for para in doc.paragraphs:
    print(para.text)
```

```
Heading for the document
Your paragraph goes here, hey there, bold here, and these words are italicEnd of the Paragraph1
2nd Paragraph
2nd paragraph starts here,  and these words are regularEnd of the Paragraph 2
```

# Working with CSV files and JSON data

# Working with CSV files

- CSV stands for “comma-separated values,” and CSV files are simplified spreadsheets stored as plaintext files.
- JSON is a format that stores information as JavaScript source code in plaintext files (JSON is short for JavaScript Object Notation.)

# Reading CSV Files

```python
import csv
# opening the CSV file
with open('example1.csv', mode ='r')as file:
    csvFile = csv.reader(file)
    # displaying the contents of the CSV file
    for lines in csvFile:
        print(lines)
```

```
['4/5/2014 13:34', 'Apples', '73']
['4/5/2014 3:41', 'Cherries', '85']
['4/6/2014 12:46', 'Pears', '14']
['4/8/2014 8:59', 'Oranges', '52']
['4/10/2014 2:07', 'Apples', '152']
['4/10/2014 18:10', 'Bananas', '23']
['4/10/2014 2:40', 'Strawberries', '98']
```

# Writing to CSV Files

```python
import csv
fields = ['Name', 'Branch', 'Year', 'CGPA']

rows = [ ['Nikhil', 'COE', '2', '9.0'],
         ['Sanchit', 'COE', '2', '9.1'],
         ['Aditya', 'IT', '2', '9.3'],
         ['Sagar', 'SE', '1', '9.5'],
         ['Prateek', 'MCE', '3', '7.8'],
         ['Sahil', 'EP', '2', '9.1']]

# name of csv file
filename = "student_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
    # writing the fields
    csvwriter.writerow(fields)
    # writing the data rows
    csvwriter.writerows(rows)
```

# Writing JSON to a file in Python using dumps()

- The JSON library in Python uses dump() or dumps() function to convert the Python objects into their respective JSON object, so it makes it easy to write data to files.

| PYTHON OBJECT | JSON OBJECT |
|---|---|
| Dict | object |
| list, tuple | array |
| str | string |
| int, long, float | numbers |
| True | true |
| False | false |
| None | null |

```python
import json
# Data to be written
dictionary = {
    "name": "sathiyajith",
    "rollno": 56,
    "cgpa": 8.6,
    "phonenumber": "9976770500"
}

# Serializing json
json_object = json.dumps(dictionary, indent=4)

# Writing to sample.json
with open("sample.json", "w") as outfile:
    outfile.write(json_object)
```

sample - Notepad

File Edit Format View Help

```
{
    "name": "sathiyajith",
    "rollno": 56,
    "cgpa": 8.6,
    "phonenumber": "9976770500
}
```

Ln 1, Col 1 | 130% | Windows (CRLF) | UTF-8

# Reading JSON from a file using  json.load()

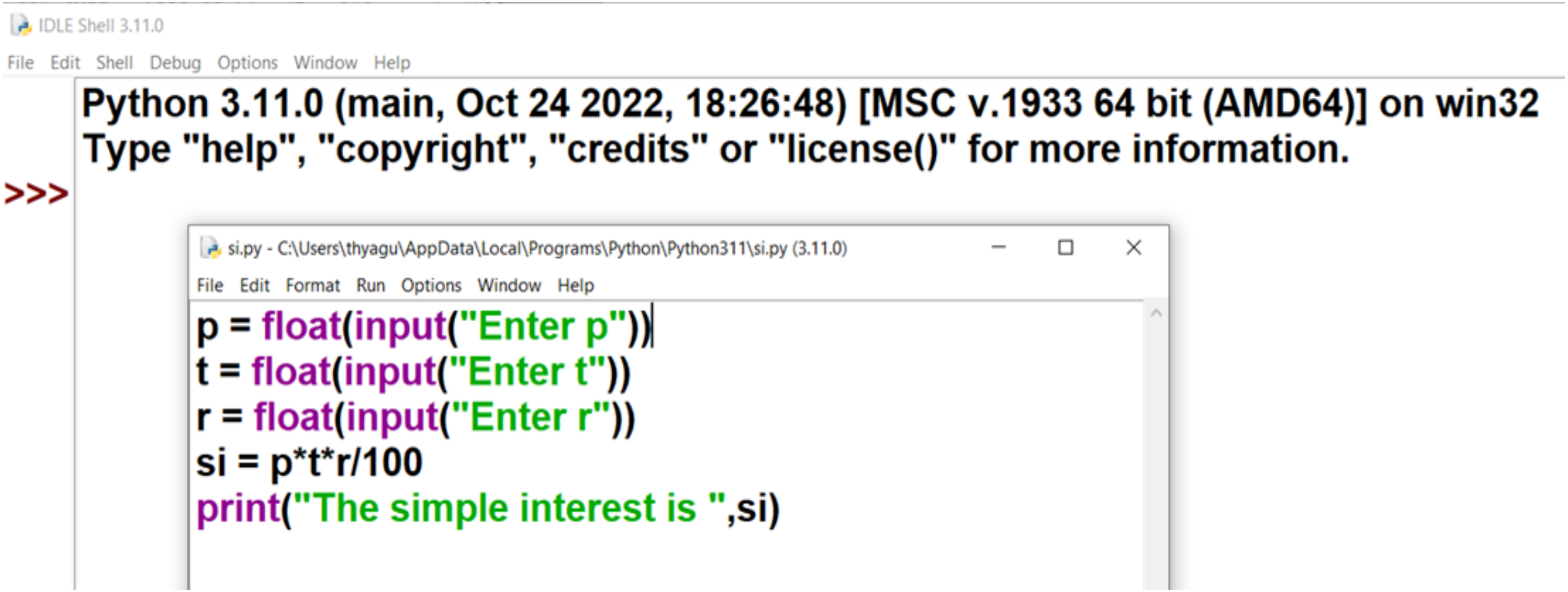
```python
import json
# Opening JSON file
with open('sample.json', 'r') as openfile:
    # Reading from json file
    json_object = json.load(openfile)

print(json_object)
print(type(json_object))
```

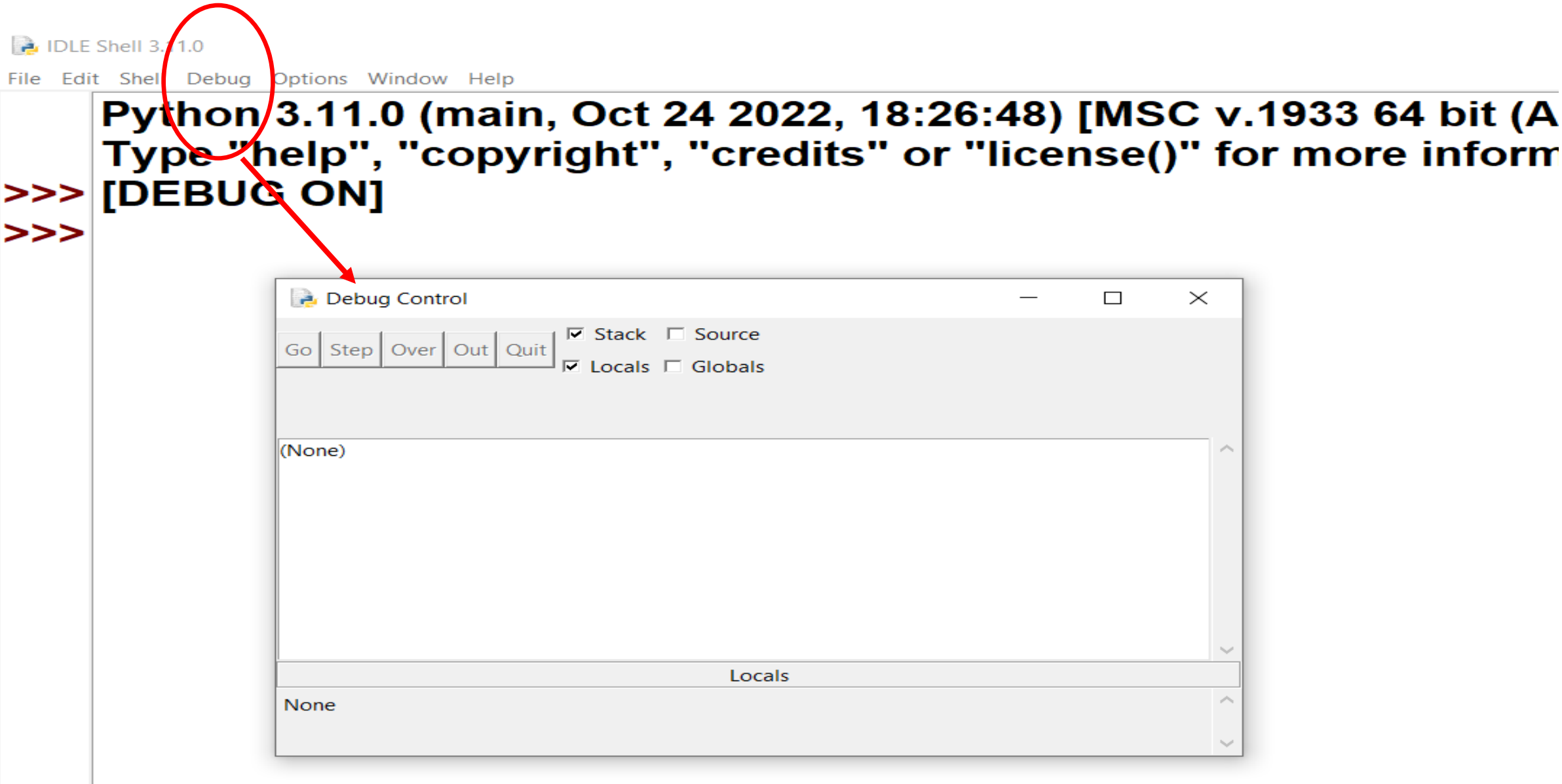
```
{'name': 'sathiyajith', 'rollno': 56, 'cgpa': 8.6, 'phonenumber': '9976770500'}
<class 'dict'>
```

# IDLE DEBUGGER

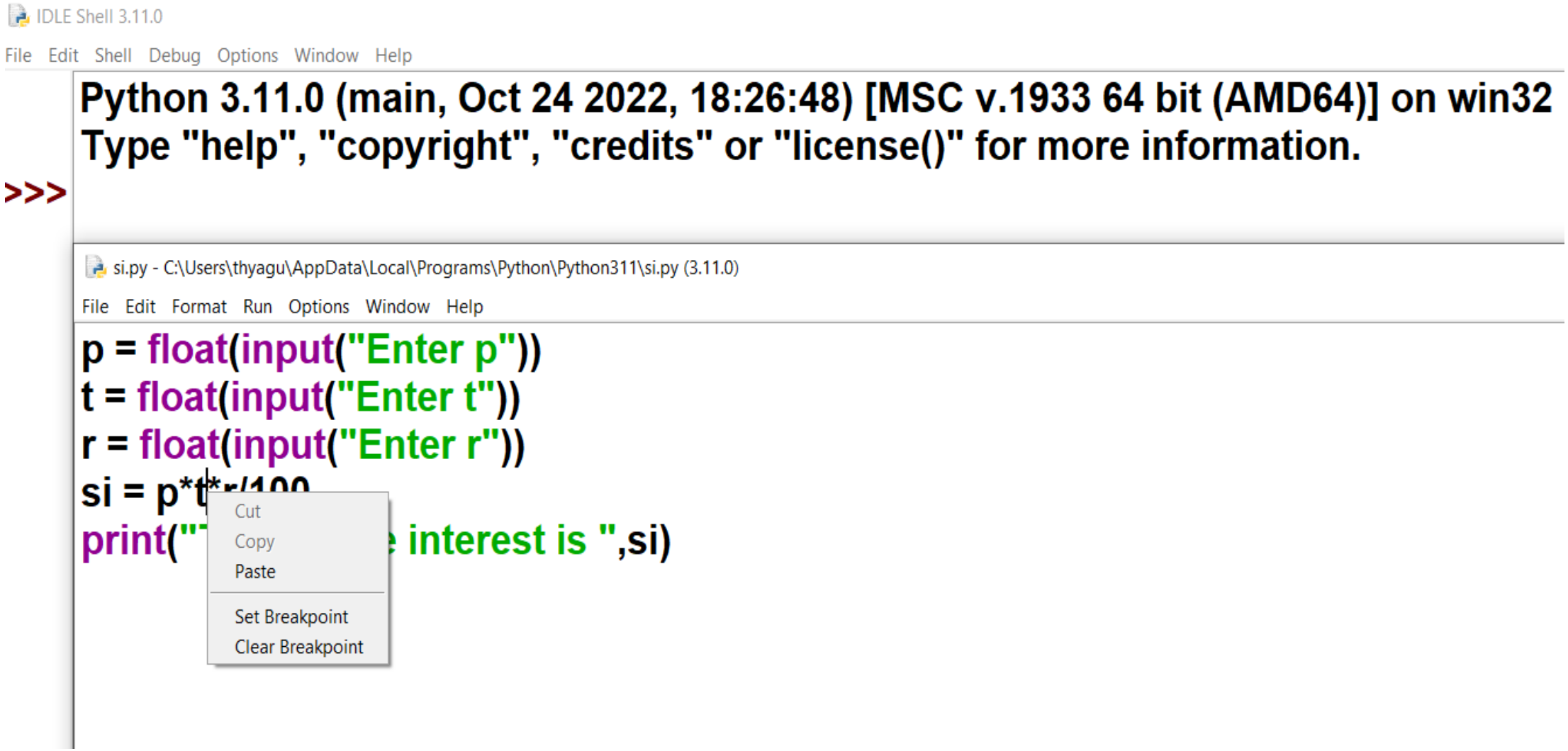
IDLE has a debugger built into it. It is very useful for stepping through a program and watching the variables change values. Start IDLE and open the program source file as illustrated below:



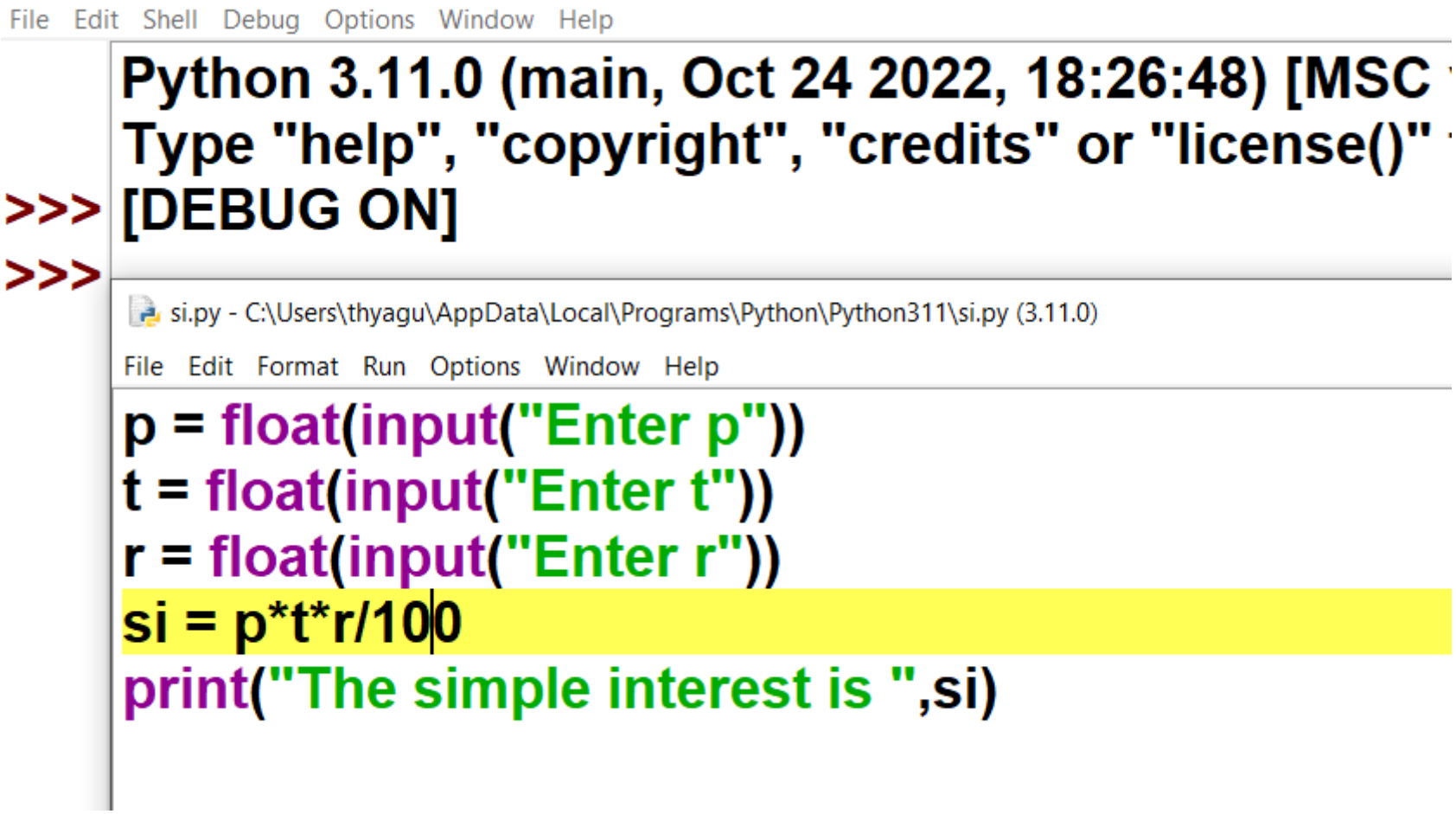
**In the Shell window, click on the Debug menu option at the top and then on Debugger. You will see a "Debug Control" window like this**



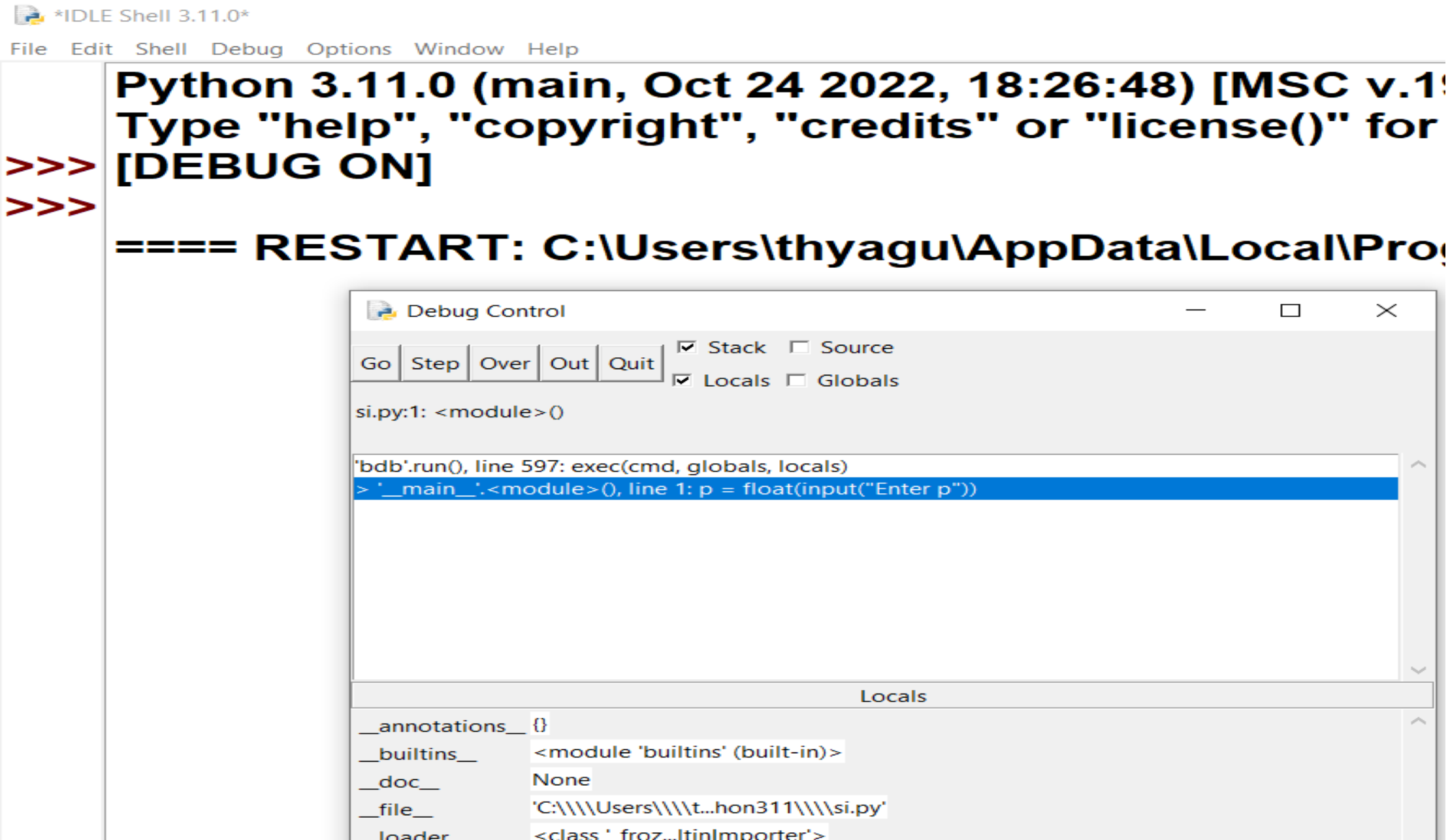
**For the debugger to be most useful, you need to set a breakpoint in your source code before you start running the program.** RIGHT click on a line of your source and choose "set breakpoint".



The background of the line you click on turns yellow to show the line marked with the breakpoint



Now run the program with F5 as usual.



*IDLE Shell 3.11.0*

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933
Type "help", "copyright", "credits" or "license()" for mor
>>> [DEBUG ON]
>>>

==== RESTART: C:\Users\thyagu\AppData\Local\Progran
Enter p
```

Debug Control

Go | Step | Over | Out | Quit

☑ Stack ☐ Source

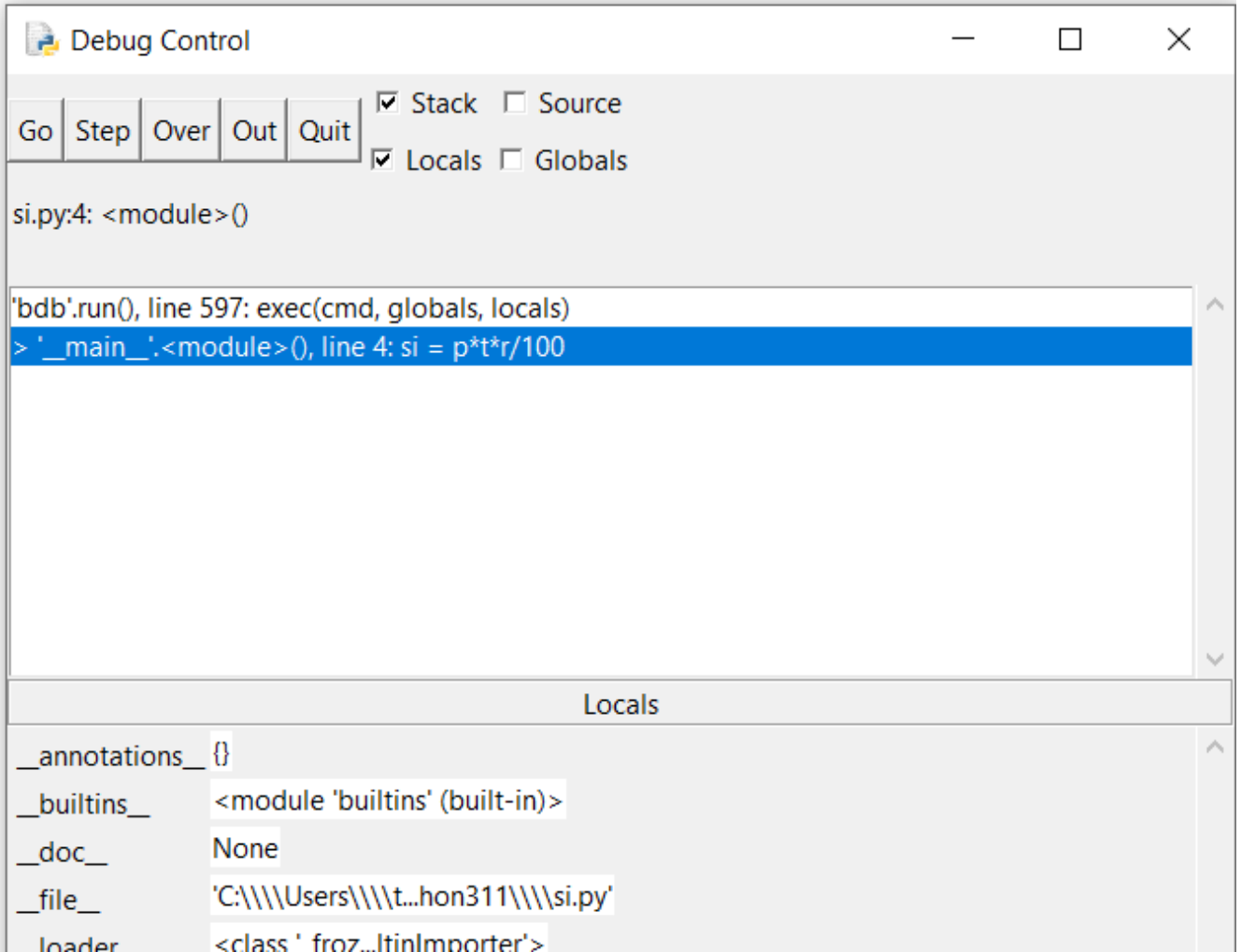
☑ Locals ☐ Globals

```
'bdb'.run(), line 597: exec(cmd, globals, locals)
> '__main__'.<module>(), line 1: p = float(input("Enter p"))
```

Locals

| | |
|---|---|
| __annotations__ | {} |
| __builtins__ | <module 'builtins' (built-in)> |
| __doc__ | None |

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1
Type "help", "copyright", "credits" or "license()" for
>>> [DEBUG ON]
>>>
==== RESTART: C:\Users\thyagu\AppData\Local\Pro
Enter p3
Enter t2
Enter r3
```



Debug Control

Go | Step | Over | Out | Quit

☑ Stack ☐ Source

☑ Locals ☐ Globals

si.py:4: <module>()

'bdb'.run(), line 597: exec(cmd, globals, locals)

> '__main__'.<module>(), line 4: si = p*t*r/100

Locals

| | |
|---|---|
| __annotations__ | {} |
| __builtins__ | <module 'builtins' (built-in)> |
| __doc__ | None |
| __file__ | 'C:\\\\Users\\\\t...hon311\\\\si.py' |
| __loader__ | <class '_froz...ltinImporter'> |
| __name__ | '__main__' |
| __package__ | None |
| __spec__ | None |
| p | 3.0 |
| r | 3.0 |
| t | 2.0 |

Locals

| | |
|---|---|
| __annotations__ | {} |
| __builtins__ | <module 'builtins' (built-in)> |
| __doc__ | None |
| __file__ | 'C:\\\\Users\\\\t...hon311\\\\si.py' |
| __loader__ | <class '_froz...ltinImporter'> |
| __name__ | '__main__' |
| __package__ | None |
| __spec__ | None |
| p | 3.0 |
| r | 3.0 |
| t | 2.0 |

*IDLE Shell 3.11.0*

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MS
Type "help", "copyright", "credits" or "license(
>>> [DEBUG ON]
>>> [DEBUG OF
>>> [DEBUG ON
>>>

==== RESTA
Enter p 2
Enter t 3
Enter r 2
```

si.py - C:\Users\thyagu\AppData\Local\Programs\Python\Python311\si.py (3.1

File Edit Format Run Options Window Help

```
p = float(input("Enter p"))
t = float(input("Enter t"))
r = float(input("Enter r"))
si = p*t*r/100
print("The simple interest is ",si)
```

Debug Control

Go Step Over Out Quit

☑ Stack ☐ Source

☑ Locals ☐ Globals

si.py:5: <module>()

```
'bdb'.run(), line 597: exec(cmd, globals, locals)
> '__main__'.<module>(), line 5: print("The simple interest is ",si)
```

Locals

| | |
|---|---|
| __annotations__ | {} |
| __builtins__ | <module 'builtins' (built-in)> |
| __doc__ | None |
| __file__ | 'C:\\\\Users\\\\t...hon311\\\\si.py' |
| __loader__ | <class '_froz...ltinImporter'> |
| __name__ | '__main__' |
| __package__ | None |
| __spec__ | None |
| p | 2.0 |
| r | 2.0 |
| si | 0.12 |
| t | 3.0 |

# End of Module 5