

# INTRODUCTION TO Programming USING **PYTHON**

Hello!

**I AM James Barrett**

You can find me at:

James.Barrett@nln.ie

# **Chapter - 1**

# Language Fundamentals

About the Founder,  
Versions Python, IDE's

# Introduction to Python

- Python is a very *simple* language and has a very straightforward syntax. It helps programmers to program without **Boilerplate** (prepared) code.
- It has efficient high-level data structures and an **Elegant syntax** and **Dynamic typing**, together with its **Interpreted** nature, makes it an ideal language for scripting and rapid application development in many areas on most platforms.

## JAVA

```
public class AddTwoIntegers {  
    public static void main(String[] args) {  
  
        int a = 10;  
        int b = 20;  
  
        int sum = a + b;  
  
        System.out.println("The sum is: " + sum);  
    }  
}
```

## PYTHON

```
a = 10  
b = 20  
sum = a + b  
print("The sum is:", sum)
```

# About the Founder

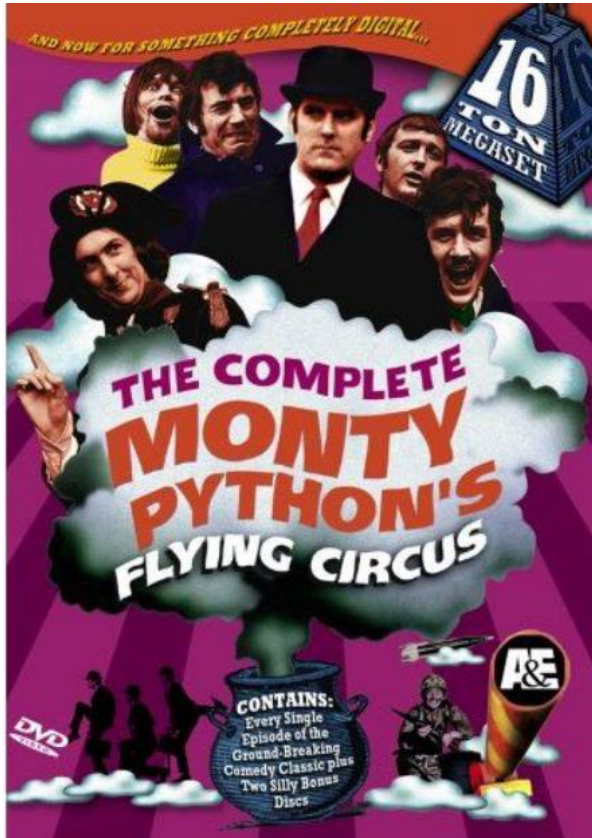
- Python was developed by **Guido Van Rossum** in 1989 while working at National Research Institute at Netherlands.
- But officially Python was made available to public in 1991. With the official Date of Birth for Python is Feb 20th, 1991.



“Python is an experiment on how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.”

*Guido Van Rossum*

# Fun fact – About the name

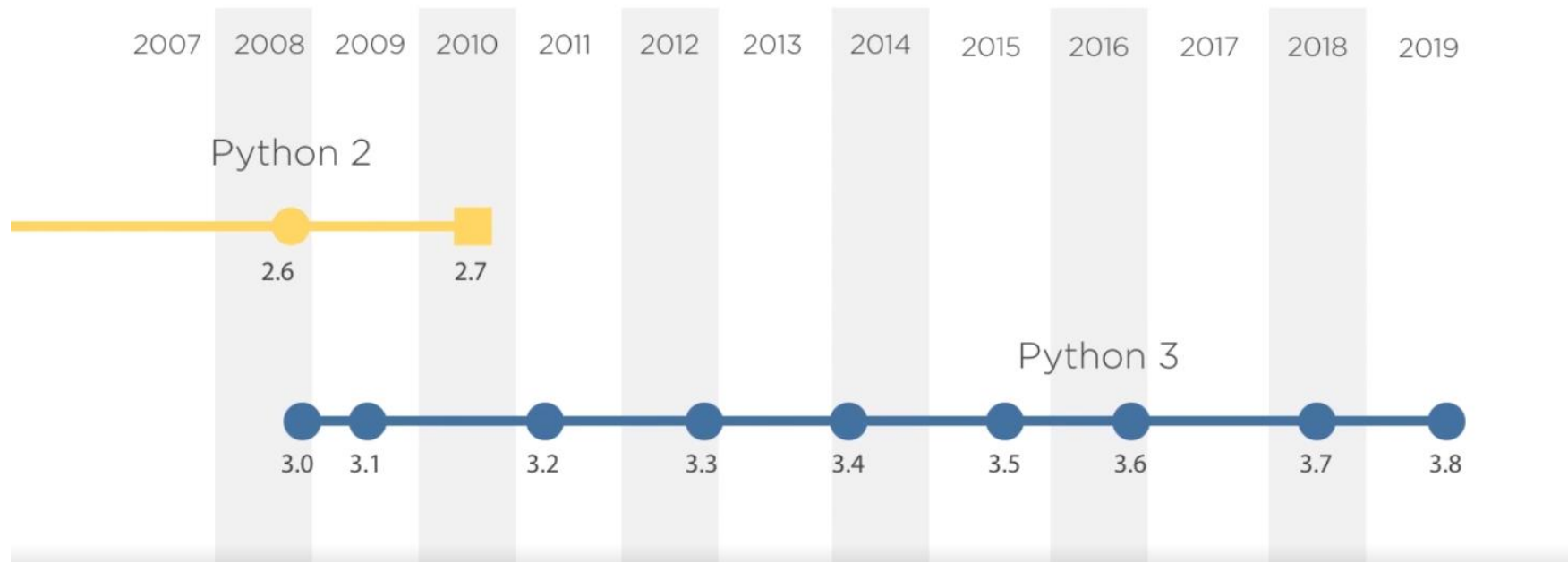


- The name Python was selected from the TV Show.
- "The Complete Monty Python's Flying Circus", which was broadcasted in BBC from 1969 to 1974.



# Versions of Python

## Python Release Timeline



# Versions of Python



There are two major versions of the Python language, Python 2, which is the widely-deployed legacy language, and Python 3, which is the present and future of the language.

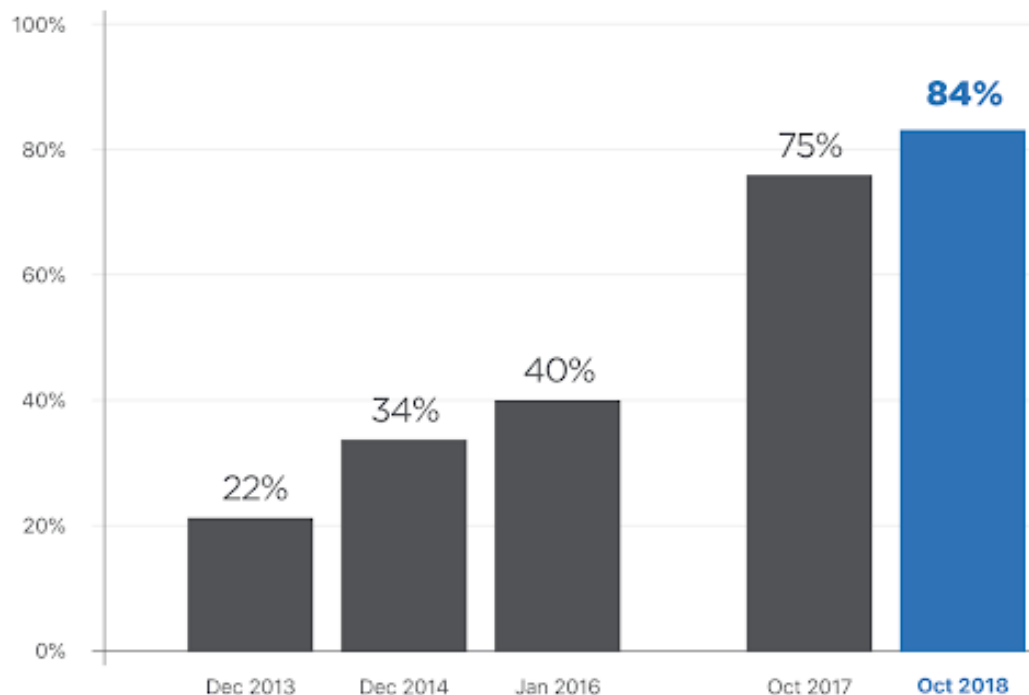


It's now over a decade since the transition from Python 2 to Python 3 was begun, Python 2 is not maintained from the year 2020.



We'll be using Python 3 for this course, and everything we show will work on Python 3.6 or later, and most things will work on Python 3.3 or later.

## Python 3 Adoption



**Note:** The adoption data for Python 3 covering previous years are derived from publicly available surveys: [2014](#), [2015](#), [2016](#) and [2017](#).

# Where is Python used?

- Python programming is a skill that can be used in virtually any industry. From industries like finance, healthcare, and insurance, to fields like aerospace to entertainment – Python is driving innovation and new solutions.
- Python has broad uses – but it's becoming more important specifically in the following areas:

•Web development

•Data Analytics

•Machine learning

•Artificial intelligence

•Task automation

•Computing security

# 7 World-Class Companies that use Python

Google

NETFLIX

 Dropbox

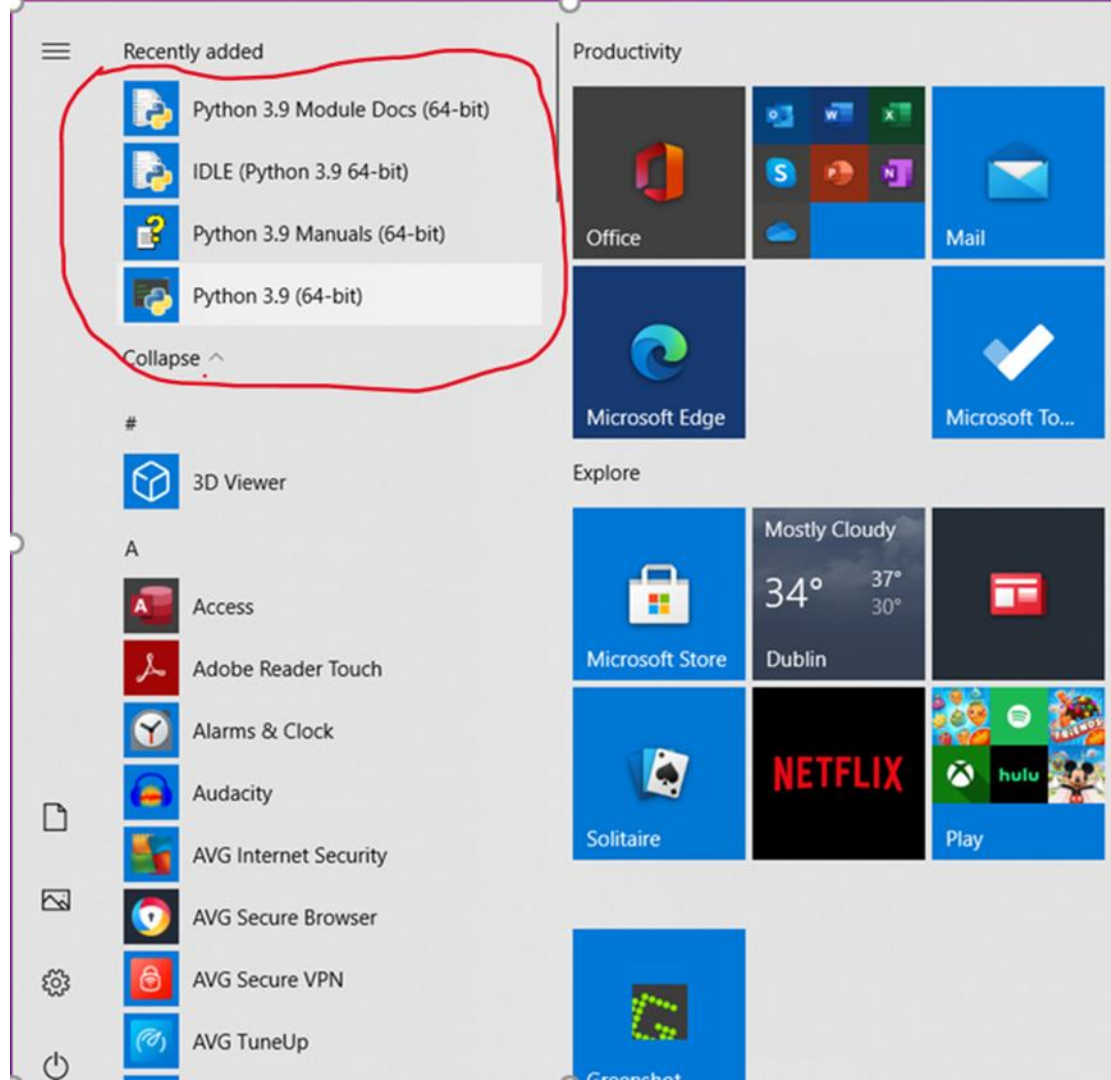
 reddit

 Instagram

stripe

 shopify

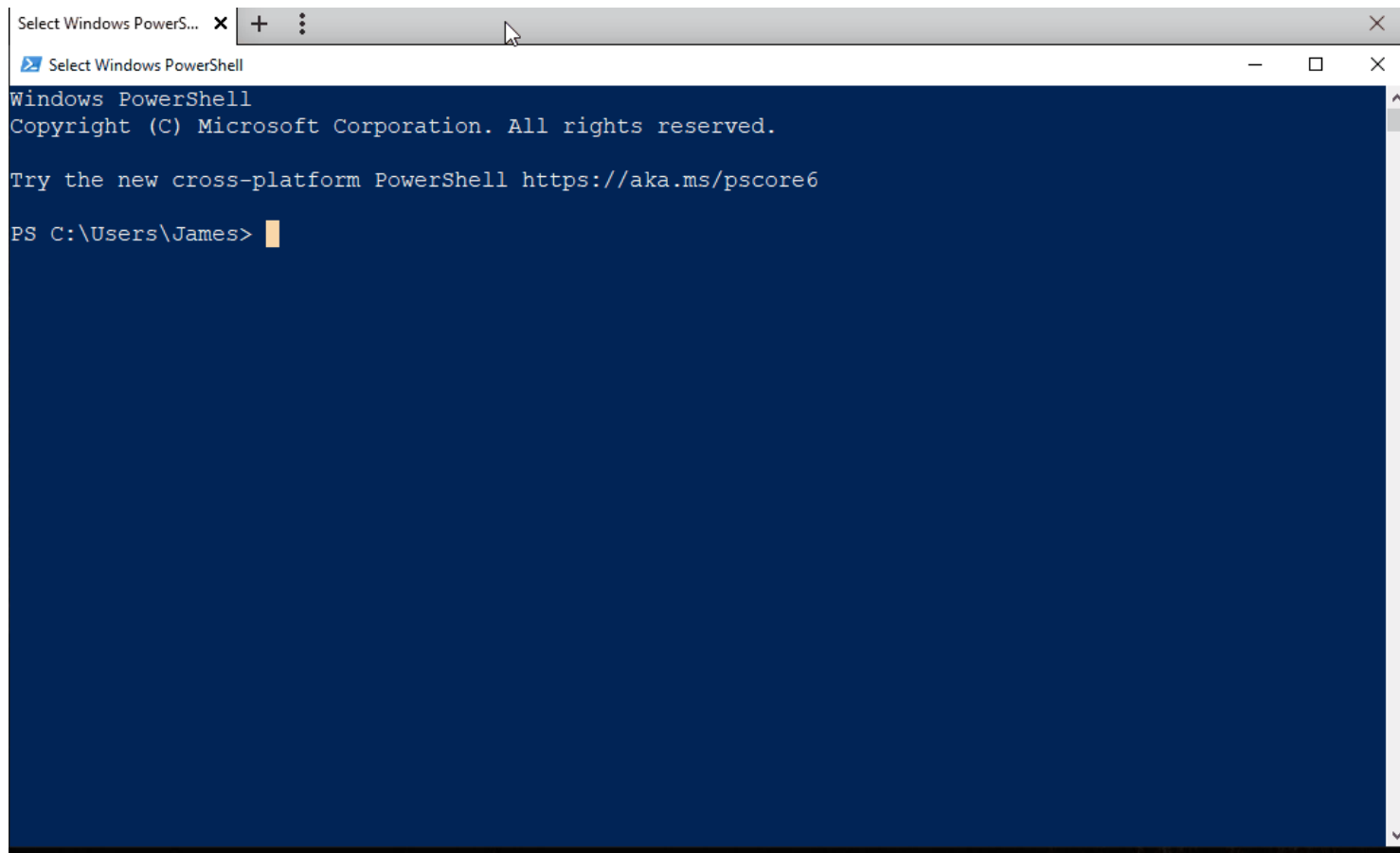
After  
Installing  
Python



# Examining the Python 3.9

- Let's examine the items in the Python 3.9 group from the bottom up.
- The **Python 3.9 Modules Docs** option provides access to all of the documentation included with all of the **installed modules** that are available within your Python system.
- The **Python 3.9 Manuals** option opens the entire set of Python language documentation in the standard Windows help utility. This material is a copy of the Python 3 documentation available on the Web.
- The **Python 3.9** option fires up a text-based interactive command prompt, `>>>`, which is used to experiment with code as you write it.
- The final option, **IDLE (Python 3.9)**, runs the Python **integrated development environment (IDE)**, which is called IDLE. This is a very simple **IDE** that provides access to Python's `>>>` prompt, a passable text editor, the Python debugger, and the Python documentation.

# Using PowerShell to start python

A screenshot of a Windows PowerShell window. The window has a title bar with the text "Select Windows PowerS..." and a tab icon. The main content area has a dark blue background with white text. The text displayed is: "Windows PowerShell", "Copyright (C) Microsoft Corporation. All rights reserved.", "Try the new cross-platform PowerShell https://aka.ms/pscore6", and "PS C:\Users\James>".

```
Select Windows PowerS... x + :  
Select Windows PowerShell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
  
PS C:\Users\James>
```



# Running Python Programs

Once you finish the installation process, you can run Python.


- **1. Run Python in Immediate mode**
- **2. Run Python in the Integrated Development Environment (IDE)**

# 1. Run Python in Immediate/Interactive mode

- Once Python is installed, typing `python` in the command line will invoke the interpreter in immediate/interactive mode. We can directly type in Python code, and press Enter to get the output.
- When commands are read from a shell, the interpreter is said to be in interactive mode.
- In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (`>>>`); for continuation lines it prompts with the secondary prompt, by default three dots (`...`).

# Interactive Mode

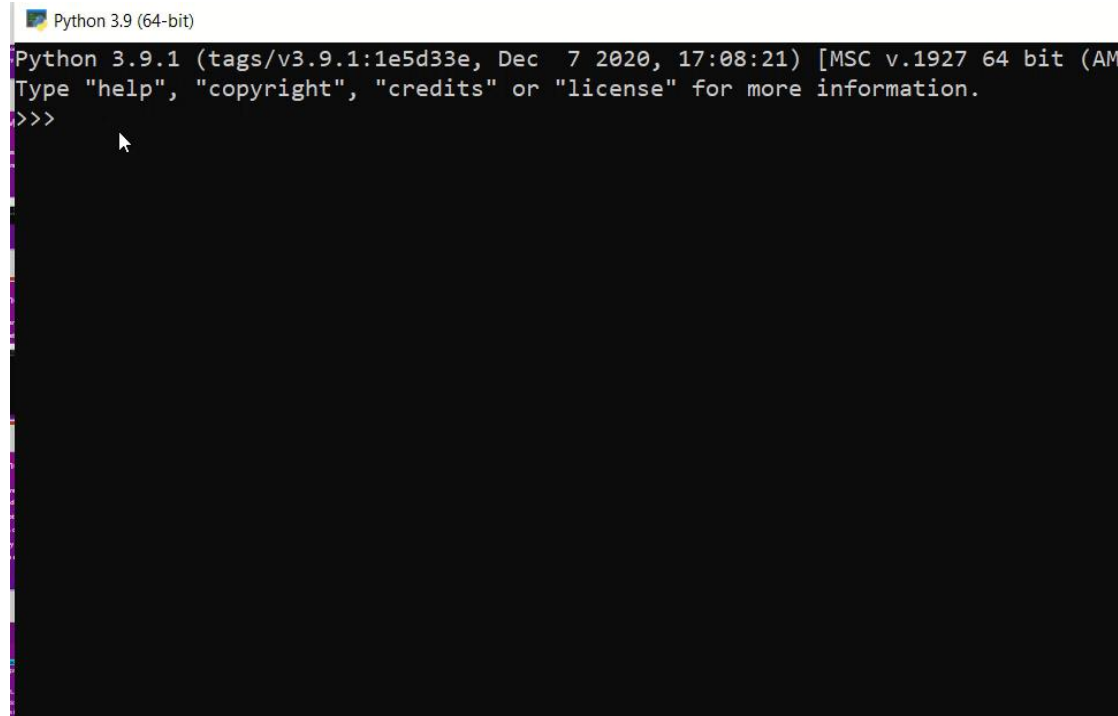
- The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

 Python 3.9 (64-bit)

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

# Interactive mode

- Continuation lines are needed when entering a multi-line construct. As an example, look at this if statement:

A screenshot of a Python 3.9.1 (64-bit) interactive shell window. The window title is "Python 3.9 (64-bit)". The text inside the shell reads: "Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]>>>". The prompt ">>>" is followed by a mouse cursor. The background is black, and the text is white. The window has a standard Windows-style title bar and a vertical scrollbar on the left side.

```
Python 3.9 (64-bit)
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]>>>
```

# Interactive mode

- In the above example, input and output are distinguished by the presence or absence of prompts (>>> and ...):
- You must type everything after the prompt, when the prompt appears.
- 
- Lines that do not begin with a prompt are output from the interpreter.
- Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

# REPL

- ▶ This Python command-line environment is a **Read, Eval, Print Loop**. Python will read whatever input we type in, evaluate it, print the result, and then loop back to the beginning.
- ▶ You'll often hear it referred to as simply the REPL.
- ▶ When started, the REPL will print some information about the version of Python you're running, and then it will give you a triple-arrow prompt. This prompt tells you that Python is waiting for you to type something. Within an interactive Python session, you can enter fragments of Python programs and see instant results.

# Using Python as a Calculator



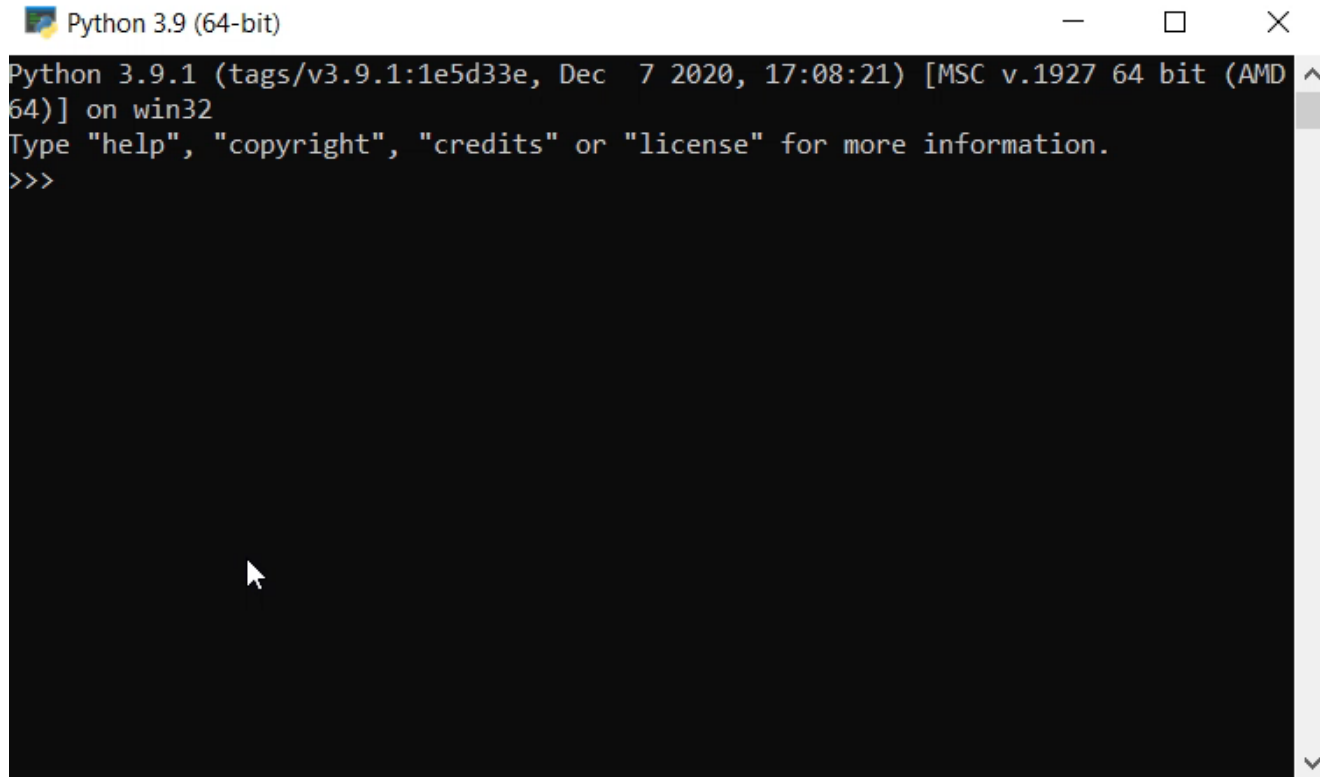
Let's try some simple Python commands.



The interpreter acts as a simple calculator: you can type an expression at it, and it will write the value.



Expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work just like in most other languages.

A screenshot of a Windows command prompt window titled "Python 3.9 (64-bit)". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The text inside the window is as follows:

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD  
64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

A mouse cursor is visible in the lower-left area of the black terminal window.

Besides numbers, Python can also manipulate strings, which can be expressed in several ways.  
We'll however learn that in the coming sessions...



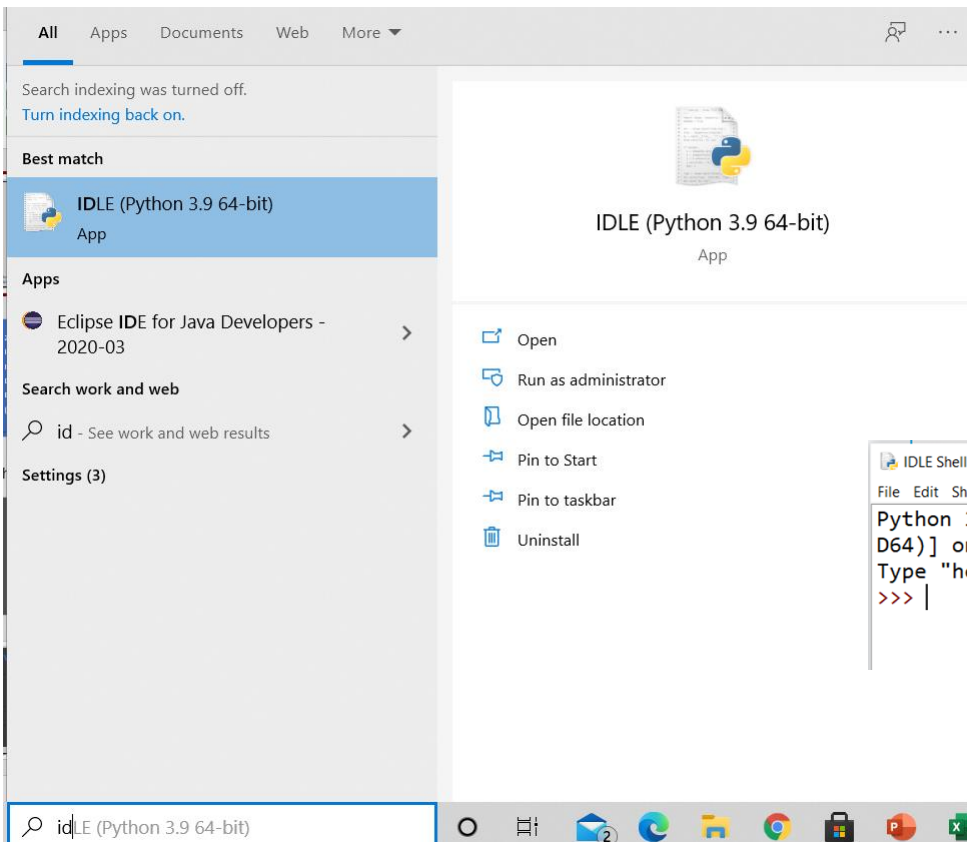
# Exercise - 1

Start the Python interpreter and use it as a calculator.

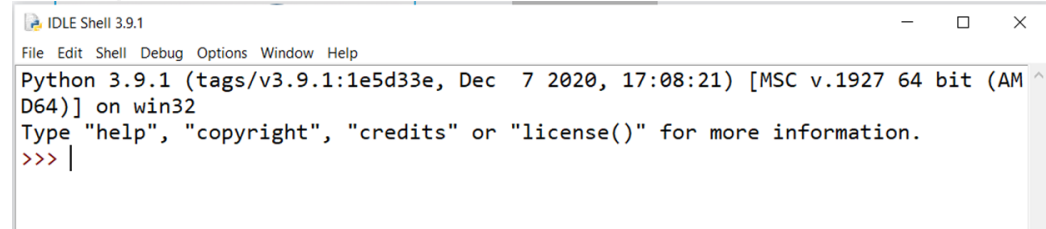
1. How many seconds are there in 42 minutes 42 seconds?
2. How many miles are there in 10 kilometres? Hint: there are 1.61 kilometres in a mile.
3. Assuming you start with €100 and earn 5% interest each year, how much will you have at the end of one year, two years and three years?

## 2. Run Python in the Integrated Development Environment (IDE)

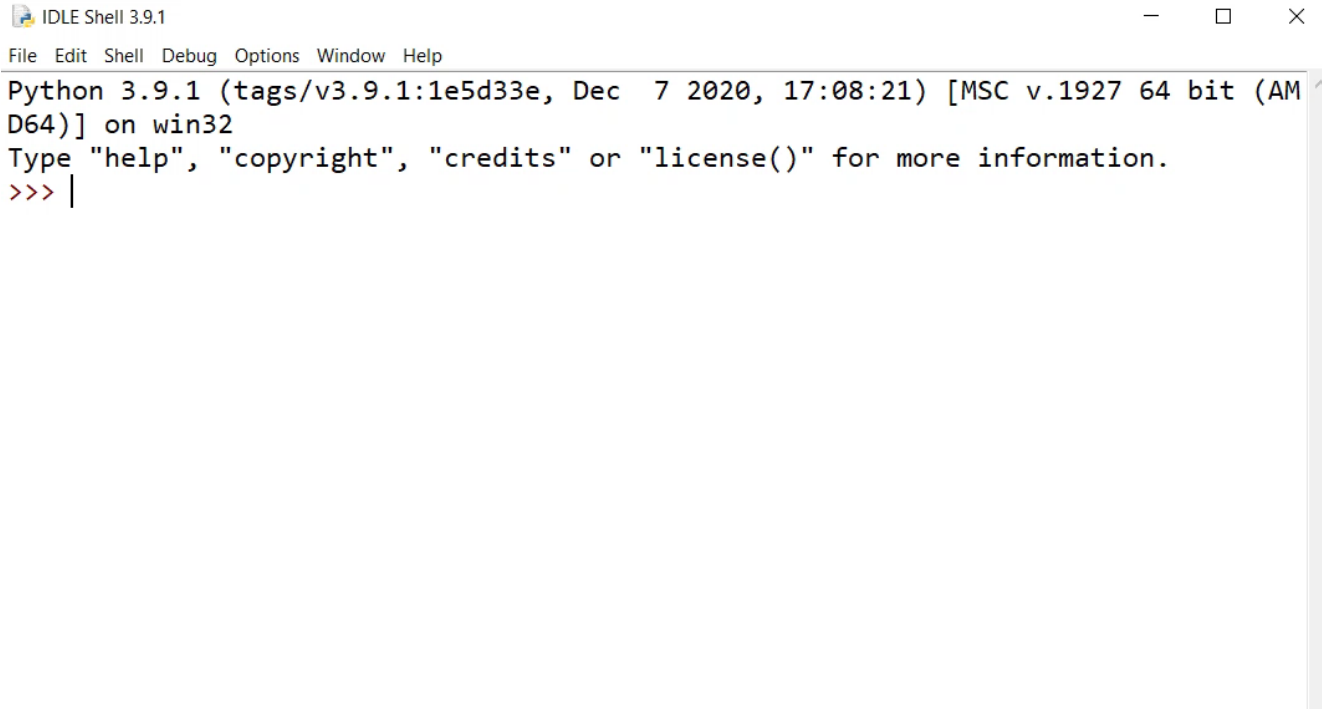
- We can use any text editing software to write a Python script file.
- We just need to save it with the .py extension. But using an IDE can make our life a lot easier. IDE is a piece of software that provides useful features like code hinting, syntax highlighting and checking etc. to the programmer for application development.
- By the way, when you install Python, an IDE named IDLE is also installed. You can use it to run Python on your computer. It's a decent IDE for beginners.



When you open IDLE, an interactive Python Shell is opened.

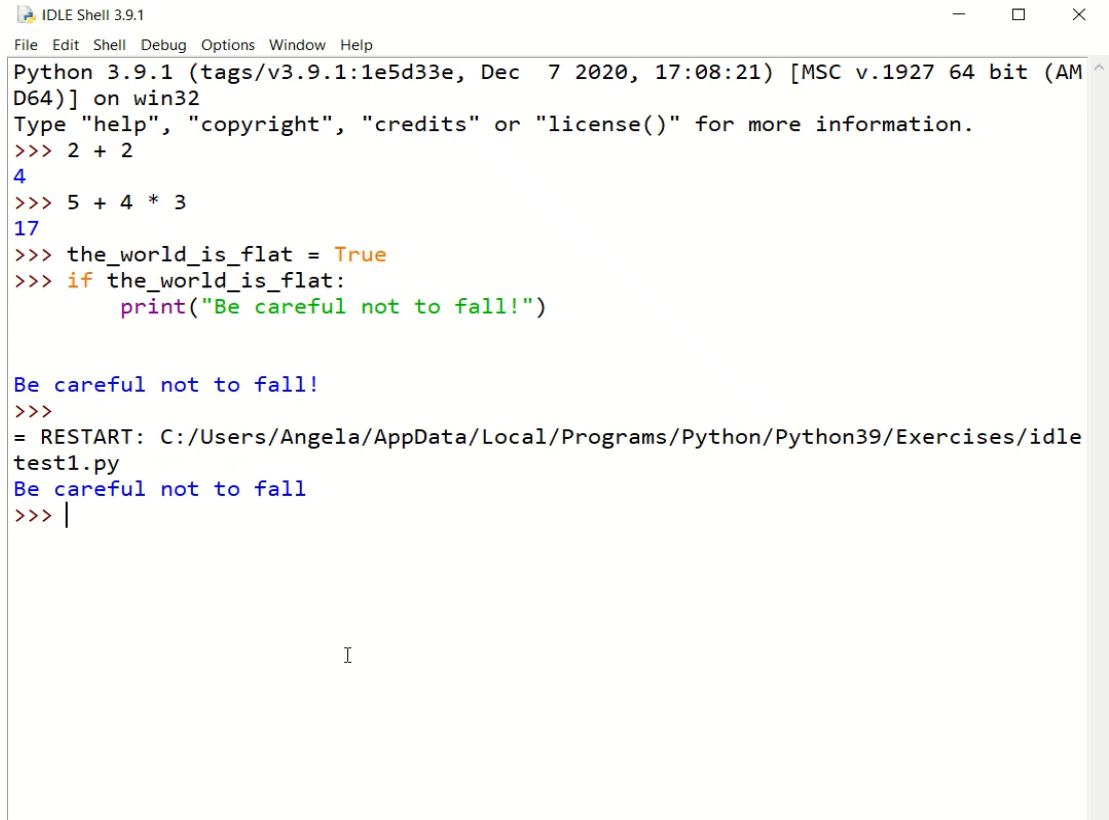


# Working with Python IDLE



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

# Creating a new python file

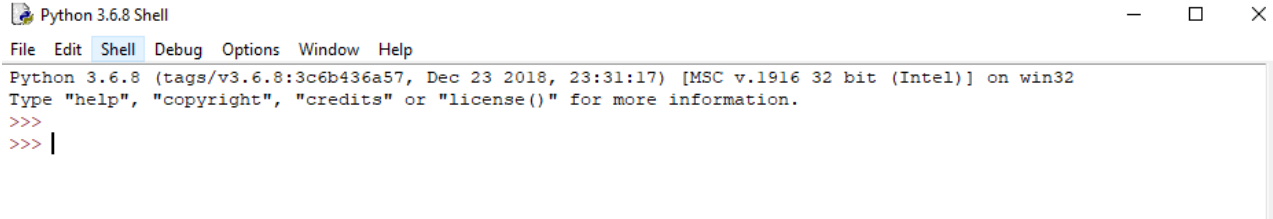


```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 2 + 2
4
>>> 5 + 4 * 3
17
>>> the_world_is_flat = True
>>> if the_world_is_flat:
    print("Be careful not to fall!")

Be careful not to fall!
>>>
= RESTART: C:/Users/Angela/AppData/Local/Programs/Python/Python39/Exercises/idle
test1.py
Be careful not to fall
>>> |
```

I

a. Open IDLE. It will look like this:

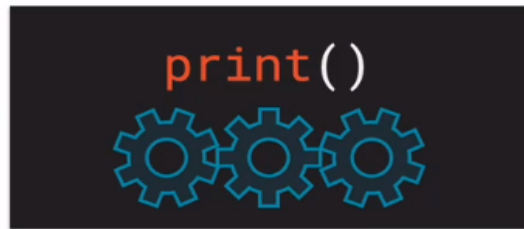
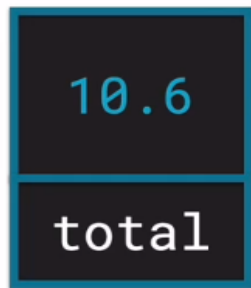
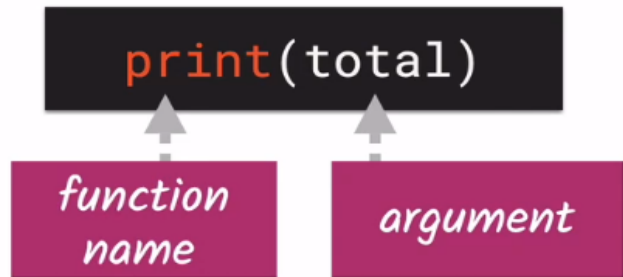


b. At the three chevrons (>>>), type the following line exactly and press enter:  
`print ("Hello world ")`

Congratulations, you have just written your first line of code!

c. Create a folder called “PythonScripts” on your Desktop. This will be your code repository.

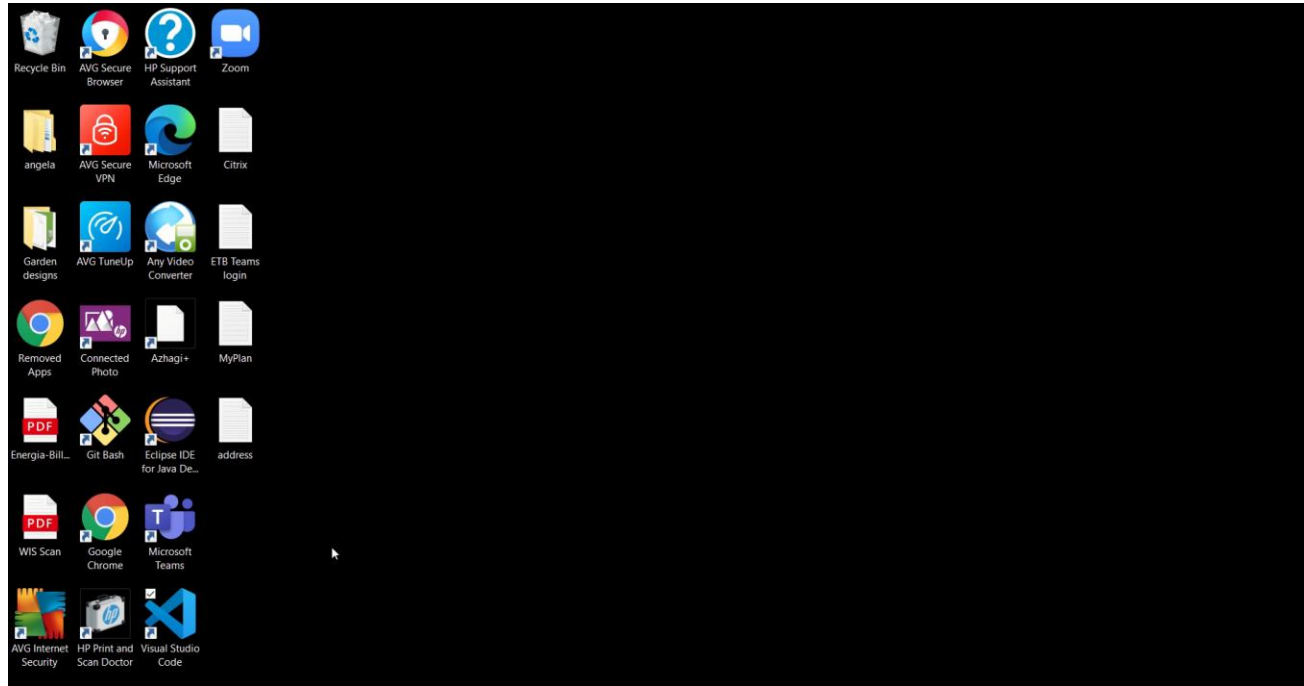
d. Back in IDLE, click on the File tab at the top of the window, then the New File option. The new window that opens is a code editor. This is where you can write programs, they are also called "scripts". Type the same line of code as in part 2b and click File > Save or Ctrl+S, (keyboard shortcuts are different on a Mac, for example Ctrl on Windows is Command on Mac, but you should be able to find the shortcut in the menu next to the option you want). In the Save window, navigate to your PythonScripts folder on your Desktop as the location to save your file. Give it the file name `hello_world.py` and click Save.



*You can think of this function as a black box machine. We don't know how it works...*

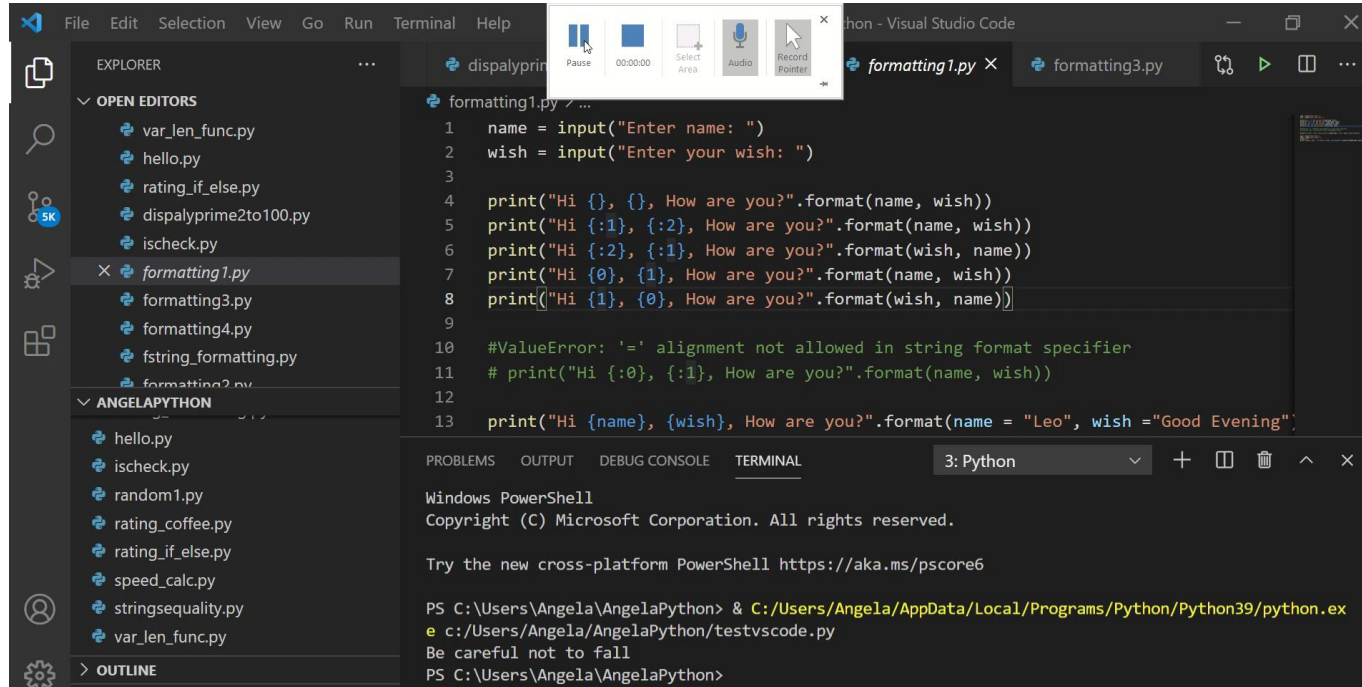
*But we do know it will output the given value to the screen.*

# Creating a new .py file in VS Code





# Creating a new file in VS Code – Another Example



▸ Variables, Constants  
and Identifiers

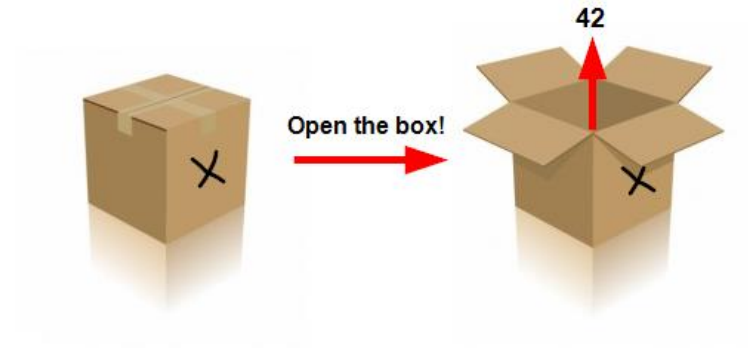
# Variables

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data that can be changed later in the program.

For example,

```
number = 42
```

Here, we have created a variable named `number`. We have assigned the value 42 to the variable.



## Assigning values to Variables in Python

- As you can see from the above example, you can use the assignment operator `=` to assign a value to a variable.
- Another great thing about Python is that the value as well as the data type of the variable can be changed during the code execution.
- As you can see in the output, from the next slide, the data type of the variable was changed in the same code execution. The Python function `"type"` is used to get the information about the data type of the variable.

```
test.py x
1 X = 22
2 print(X)
3 print(type(X))
4 X = "This is a string"
5 print(X)
6 print(type(X))
```

## Output

```
C:\Python1\Scripts\python
PyCharm Community Edition 2019.2.3
import sys; print('Python
sys.path.extend(['C:\py
platform))
PyDev console: starting.
Python 3.7.3 (v3.7.3:ef4
[MSC v.1916 64 bit (AMD64)] on win32
>>> runfile('C:/python/t
22
<class 'int'>
This is a string
<class 'str'>
```

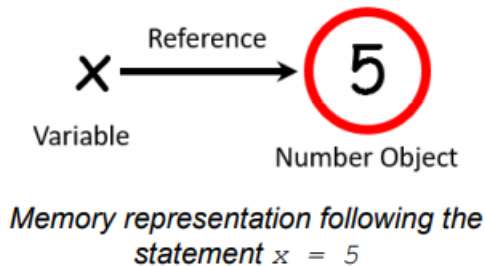
Value and the Data Type  
changes during the code  
Execution

# How is a value represented in memory?

- In Python, we don't assign values to the variables. Instead, Python gives the reference of the object(value) to the variable.
- The values are represented in memory as objects and a reference links the variable to the object.

# Memory Representation

- The diagrams below illustrate what happens when we initialise the variable `x` to 5.



- We will revisit this later, when we learn about immutability of objects

# Assigning multiple values to multiple variables

```
a, b, c = 5, 3.2, "Hello"
```

```
x = y = z = "same"
```



# Constants

- A constant is a type of variable whose value should not be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.
- For example, The number of seconds in one hour is a constant as well; it will never change: there will always be 3,600 seconds in an hour.
- Example : Declaring and assigning value to a constant
  - **PI = 3.14**
  - **GRAVITY = 9.8**
  - **MAX\_VALUE = 10**
- **Naming them in all capital letters** is a convention to separate them from variables, however, it does not actually prevent reassignment.

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

## **Rules for writing identifiers**

1. Identifiers can be a combination of let
2. In lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore \_.
3. An identifier cannot start with a digit.  
example : 1variable is invalid, but variable1 is a valid name.
4. Keywords cannot be used as identifiers.
5. We cannot use special symbols like !, @, #, \$, % etc. in our identifier.
6. An identifier can be of any length.

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are **case sensitive**.
- There are **33 keywords** in Python 3.7. This number can vary slightly over the course of time.
- All the keywords except **True, False and None** are in **lowercase** and they must be written as they are.

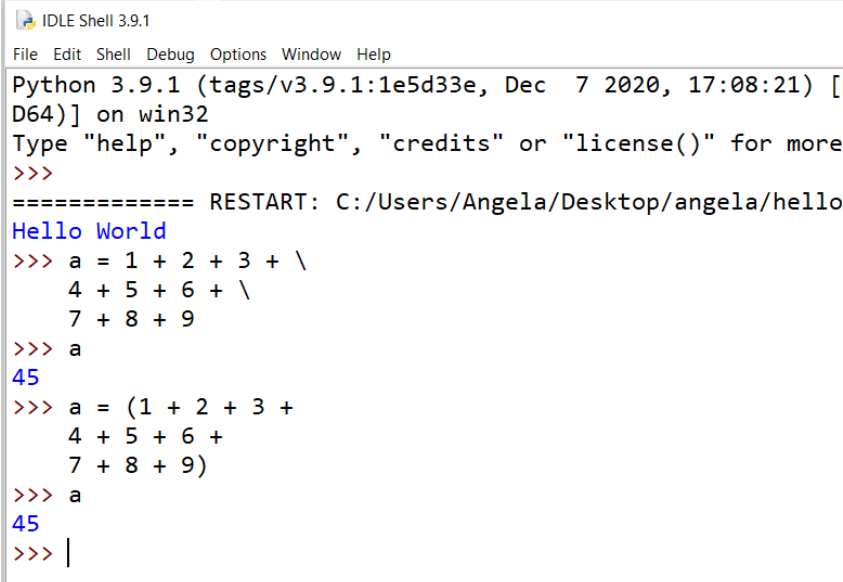
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

# Python Statement

- **Instructions that a Python interpreter can execute are called statements.**
- For example, `a = 1` is an assignment statement.
- `if` statement, `for` statement, `while` statement, etc. are other kinds of statements which will be discussed later.

# Multi-line statement

- In Python, the end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the **line continuation character** (`\`).
- This is an explicit line continuation. In Python, line continuation is implied inside parentheses `()`, brackets `[]`, and braces `{ }`
- We can also put multiple statements in a single line using semicolons, as follows:
  - `a = 1; b = 2; c = 3`

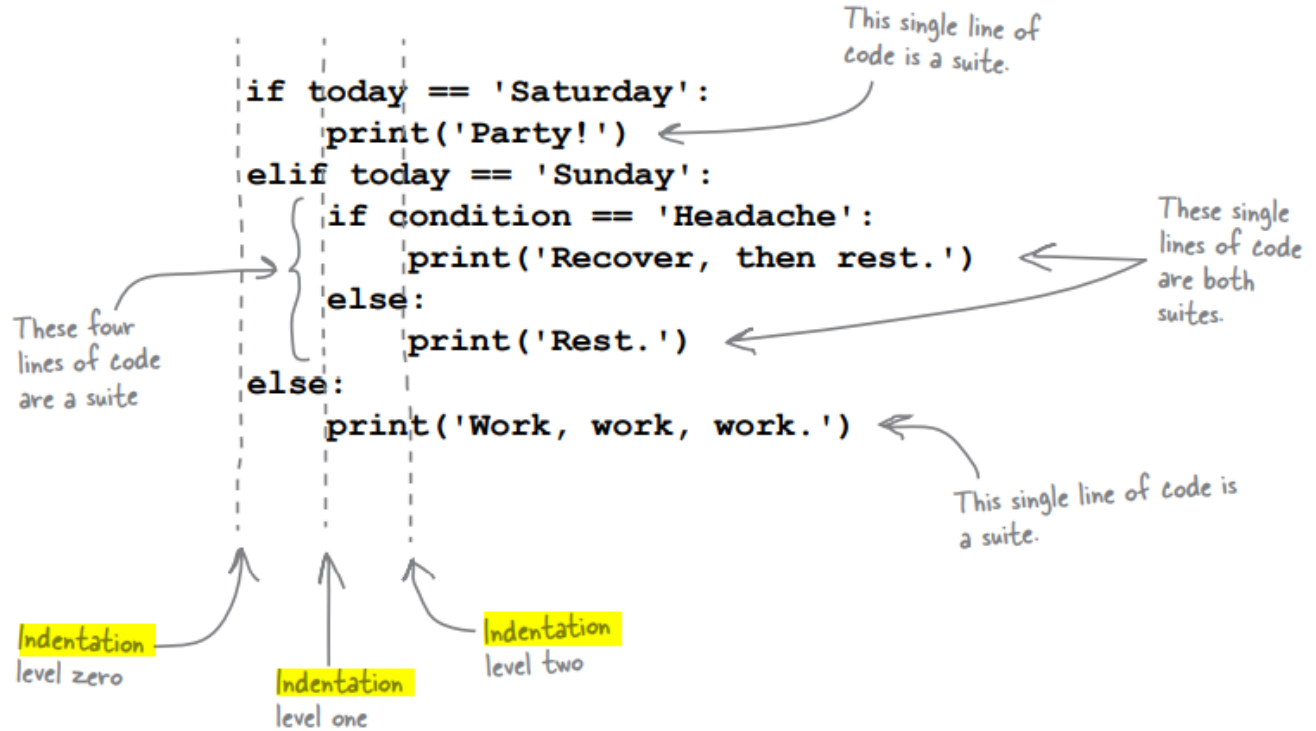


```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [D64] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: C:/Users/Angela/Desktop/angela/hello
Hello World
>>> a = 1 + 2 + 3 + \
      4 + 5 + 6 + \
      7 + 8 + 9
>>> a
45
>>> a = (1 + 2 + 3 +
      4 + 5 + 6 +
      7 + 8 + 9)
>>> a
45
>>> |
```

# Indentation

- **Indentation** refers to the spaces at the beginning of a code line.
- Where in other programming languages the **indentation** in code is for readability only, the **indentation** in **Python** is very important.
- **Python** uses **indentation** to indicate a block of code.
- Indentation refers to the **whitespaces** (usually 4 spaces or a single **tab**) that signify the beginning of a suite (block) of code. **All statements indented at the same level belong to the same suite.\*\***
- The **colon ( : )** introduces a new suite(block) of code that must be intended to right.  
The interpreter raises an error if you forgot to indent your statements after a colon.  
The end of the suite is specified by unindenting the next line of code(which is not the part of your suite).
- In Python IDLE, it automatically indents the code it takes the proper number of spaces and then you can write the code for which it will be indented.

# Indentation Example





# Comments

- Comments provide a way to leave an explanation or annotation in your source code. Comments are programmer-readable and are added to make the source code easier for a programmer to understand.
- When the Python interpreter comes across a comment, it ignores the comment and moves to the next line of code.
- There are two types of commenting features available in Python:
  - These are single-line comments and multi-line comments.

# Single- line comment

- A single-line comment begins with a **hash (#) symbol** and is useful in mentioning that the whole line should be considered as a comment until the end of the line.
- In the below example program, the first line starts with the hash symbol, so the entire line is considered a comment.
- In the second line of code, "N = 50" is a statement, and after the statement, the comment begins with the # symbol. From the # symbol to the end of this line, the line will be treated as a comment. (**inline comment**)

## Example:

```
#Defining a variable to store number.  
n = 50 #Store 50 as value into variable n.
```

## Multi-line comment

- Multi-line comment is useful when we need to comment on many lines.
- Python doesn't offer a separate way to write multiline comments.
- We can use # at the beginning of each line of comment on multiple lines.
- Here, each line is treated as a single comment and all of them are ignored.

```
# it is a  
# multiline  
# comment
```

# Write a comment

- To comment out a section of code in IDLE, highlight one or more lines to be commented and press:
  - Windows: Alt + 3
- To remove comments, highlight the commented lines and press:
  - Windows: Alt + 4

# Conventions and Pet Peeves

- According to **PEP 8**, comments should always be written in complete sentences with a single space between the # and the first word of the comment:

```
# This comment is formatted to PEP 8.
```

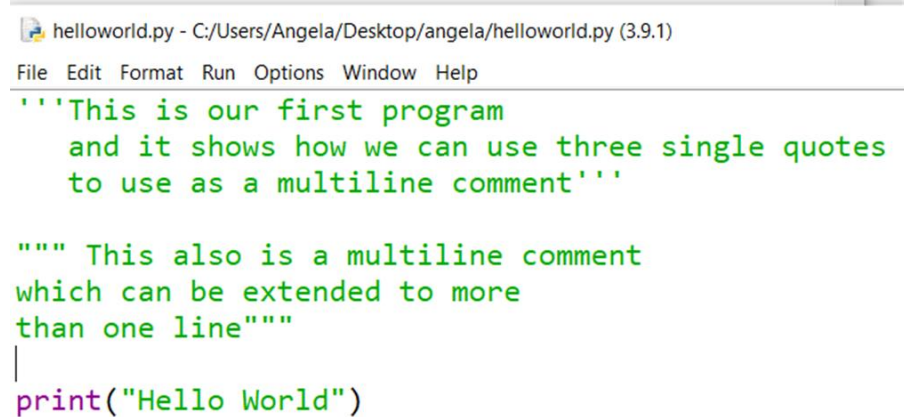
```
#this one isn't
```

- For inline comments, PEP 8 recommends at least two spaces between the code and the # symbol

```
phrase = "Hello, World" # This comment is PEP 8 compliant.  
print(phrase)# This comment isn't.
```

# Using String Literals to write Multi-line Comments

- Here, the multiline string isn't assigned to any variable, so it is ignored by the interpreter.
- Even though it is not technically a multiline comment, it can be used as one.



The screenshot shows a Python IDE window titled 'helloworld.py - C:/Users/Angela/Desktop/angela/helloworld.py (3.9.1)'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code in the editor consists of a multiline string comment enclosed in three single quotes, followed by a multiline comment enclosed in four double quotes, and finally a print statement.

```
'''This is our first program
and it shows how we can use three single quotes
to use as a multiline comment'''

""" This also is a multiline comment
which can be extended to more
than one line"""

print("Hello World")
```

# Docstrings in Python

- A docstring is short for documentation string.
- Python docstrings (documentation strings) are the string literals that appear right after the definition of a function, method, class, or module. This separates docstrings from multiline comments using triple quotes.
- Triple quotes are used while writing docstrings.

```
def double(num):  
    """Function to double the value"""  
    return 2*num
```

# Python Comments vs Docstrings

- **Python Comments**

- Comments are descriptions that help programmers better understand the intent and functionality of the program.

- **Python docstrings**

- As mentioned above, Python docstrings are strings used right after the definition of a function, method, class, or module. They are used to document our code.



# Exercise - 3

Write a program for the following

1.) Assuming you start with \$100 and earn 5% interest each year, how much will you have at the end of one year, two years and three years? Use both the IDLE and the VS Code to write the program.