

Module-1: Language Fundamentals

1.1. Introduction:

- Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.
- Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

1.2. History:

- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- We can check python version from the link: https://en.wikipedia.org/wiki/History_of_Python

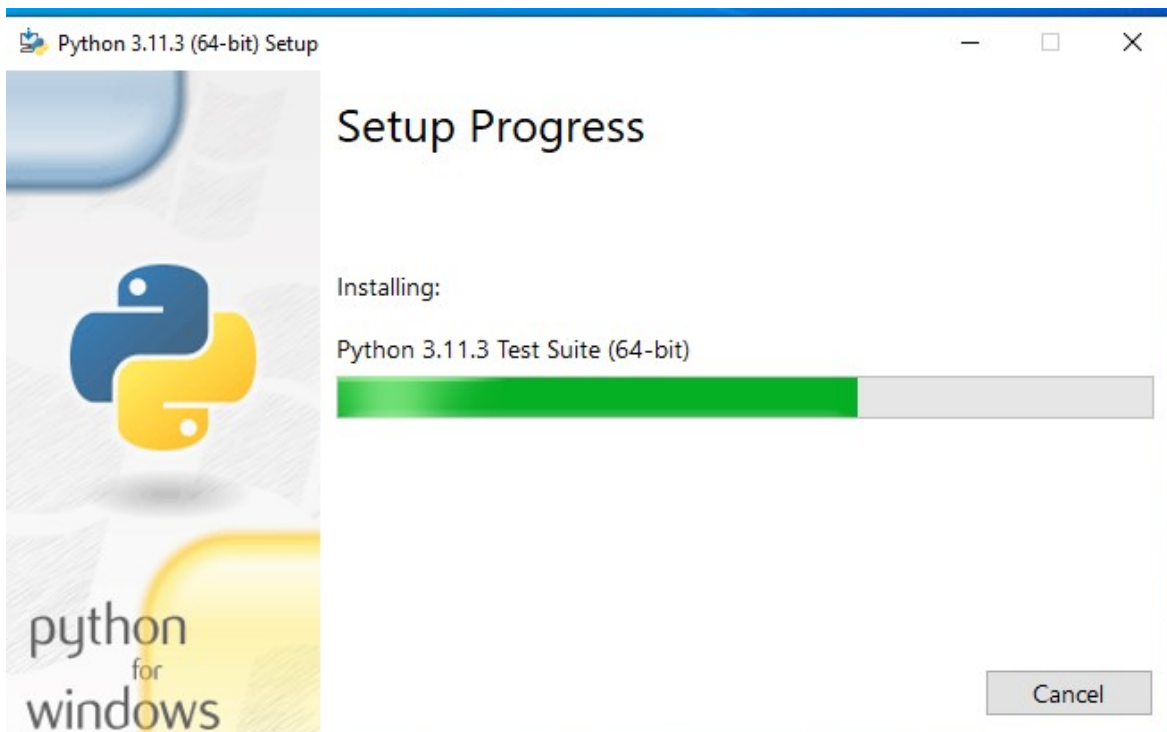
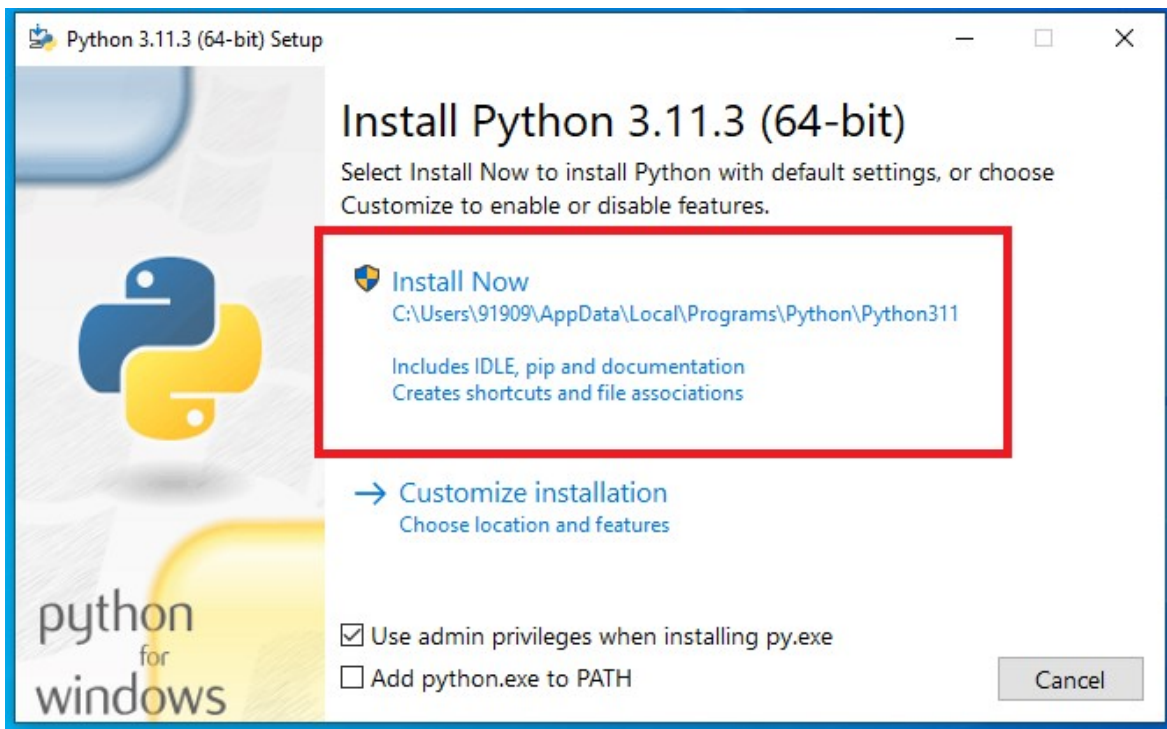
2.7	2.7.18 ^[32]	2010-07-03 ^[32]	2020-01-01 ^{[c][32]}	
3.0	3.0.1 ^[44]	2008-12-03 ^[27]	2009-06-27 ^[51]	
3.1	3.1.5 ^[52]	2009-06-27 ^[52]	2011-06-12 ^[53]	2012-04-06 ^[52]
3.2	3.2.6 ^[54]	2011-02-20 ^[54]	2013-05-13 ^{[b][54]}	2016-02-20 ^[54]
3.3	3.3.7 ^[55]	2012-09-29 ^[55]	2014-03-08 ^{[b][55]}	2017-09-29 ^[55]
3.4	3.4.10 ^[56]	2014-03-16 ^[56]	2017-08-09 ^[57]	2019-03-18 ^{[a][56]}
3.5	3.5.10 ^[58]	2015-09-13 ^[58]	2017-08-08 ^[59]	2020-09-30 ^[58]
3.6	3.6.15 ^[60]	2016-12-23 ^[60]	2018-12-24 ^{[b][60]}	2021-12-23 ^[60]
3.7	3.7.16 ^[61]	2018-06-27 ^[61]	2020-06-27 ^{[b][61]}	2023-06-27 ^[61]
3.8	3.8.16 ^[62]	2019-10-14 ^[62]	2021-05-03 ^{[b][62]}	2024-10 ^[62]
3.9	3.9.16 ^[63]	2020-10-05 ^[63]	2022-05-17 ^{[b][63]}	2025-10 ^{[63][64]}
3.10	3.10.11 ^[65]	2021-10-04 ^[65]	2023-05 ^[65]	2026-10 ^[65]
3.11	3.11.3 ^[66]	2022-10-24 ^[66]	2024-05 ^[66]	2027-10 ^[66]
3.12	3.12.0a7 ^{[67][needs update]}	2023-10-02 ^[67]	2025-05 ^[67]	2028-10 ^[67]
3.13 ^[needs update]	3.13.0 ^{[64][needs update]}	2024-10 ^{[64][needs update]}	2026-05 ^{[64][needs update]}	2029-10 ^{[64][needs update]}

1.3. Installation and Setup:

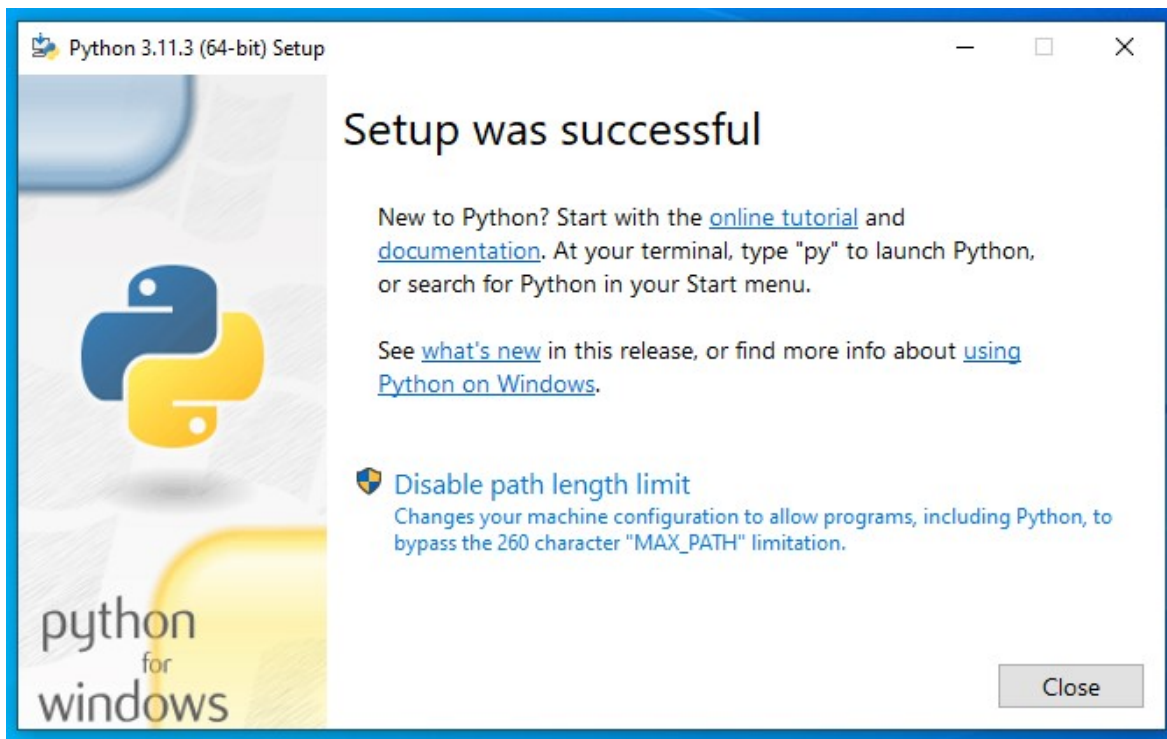
- Go to: <https://www.python.org/downloads/>
- Click on Download Python 3.11.3 button.



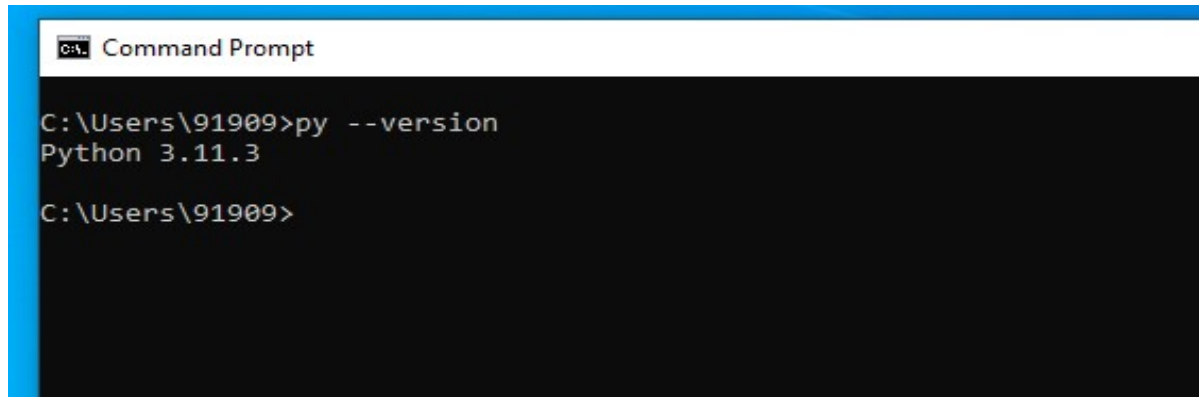
- After downloading the .exe file, install the file on your windows machine.
- Go with the default installer path.



- After successful installation, you will get something like the screenshot mentioned below.



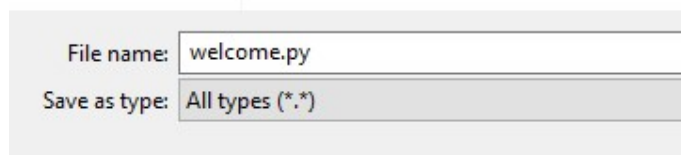
- Check python version installed:
- Open a command prompt and type **py --version**



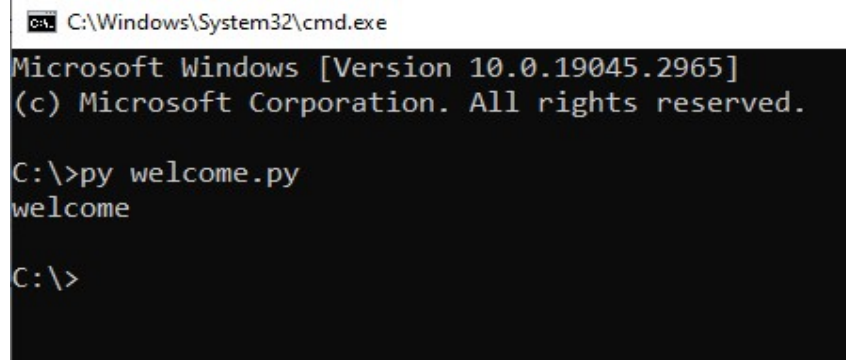
1.4 HelloWorld Example:

- Any text editor can be used to write Python programs.
- We can use Notepad, Notepad++, VSCode etc.
- Open a Notepad
- Type following line
print ('welcome')
- Save the file as: welcome.py
- Run the program in command prompt as:
py welcome.py

```
print ('welcome')
```



- The output for this Python program will be:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\>py welcome.py
welcome

C:\>
```

1.5 Features of Python:

1. Simple and Easy to Learn:

Python code looks like simple English words. There is no use of semicolons or brackets. You can tell what the code is supposed to do simply by looking at it. Learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.

2. Freeware and Open Source:

Python is developed under an OSI-approved open source license. Hence, it is completely free to use, even for commercial purposes. It doesn't cost anything to download Python or to include it in your application. It can also be freely modified and re-distributed.

3. Expressive:

Python needs to use only a few lines of code to perform complex tasks. For example, to display Hello World, you simply need to type one line - `print("Hello World")`. Other languages like Java or C would take up multiple lines to execute this.

4. Platform Independent, Portable:

Python is portable in the sense that the same code can be used on different machines. Suppose you write a Python code on a Mac. If you want to run it on Windows or Linux later, you don't have to make any changes to it. As such, there is no need to write a program multiple times for several platforms.

5. Dynamically Typed:

Many programming languages need to declare the type of the variable before runtime. With Python, the type of the variable can be decided during runtime. This makes Python a dynamically typed language. For example, if you have to assign an integer value 15 to a variable x, you don't need to write `int x = 15`. You just have to write `x = 15`.

6. Object Oriented:

A programming language is object-oriented if it focuses design around data and objects, rather than functions and logic. On the contrary, a programming language is procedure-oriented if it focuses more on functions (code that can be reused). One of the critical Python features is that it supports both object-oriented and procedure-oriented programming.

7. Extensible Language:

A programming language is said to be extensible if it can be extended to other languages. Python code can also be written in other languages like C++, making it a highly extensible language.

8. Interpreted:

When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. Programming languages such as C++ or Java are not interpreted, and hence need to be compiled first to run them. There is no need to compile Python because it is processed at runtime by the interpreter.

9. Functional Programming:

In functional programming, a program consists entirely of evaluation of pure functions. Python supports functional programming but contains features of other programming models as well.

10. Extensive Library:

Python has an extensive standard library available for anyone to use. This means that programmers don't have to write their code for every single thing unlike other programming languages. There are libraries for image manipulation, databases, unit-testing, expressions and a lot of other functionalities.

1.6 Identifiers: An identifier is a user-defined name given to identities like class, functions, variables, or any other object in Python.

Rules for writing identifiers:

- The identifiers can be a combination of characters, digits and underscore.
- It can include letters in lowercase (a-z), letters in uppercase (A-Z), digits (0-9), and an underscore (_).
- An identifier cannot begin with a digit, if it starts with a digit, it will give a Syntax error.
- Identifiers are case sensitive.
- There is no length limit for identifiers.
- Reserved words cannot be used as identifiers.

1.7 Keywords:

- Keywords are predefined, reserved words used in Python programming that have special meanings to the compiler.
- 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.
- All the keywords except True, False and None are in lowercase and they must be written as they are.
- Refer example: res_word.py

Example of reserved words in Python:

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

1.8 Data types: Data types specify the type of data that can be stored inside a variable. Python has the following data types built-in by default:

Numeric Types	int, float, complex
Text Type	str
Boolean Type	bool
Collection Types	list, tuple, set, frozenset, range
Mapping Type	dict
Binary Types	bytes, bytearray
None Type	None

Refer Example: variable_details.py

We can get the data type of any object by using the type() function:

Example:

```
x = 5.2
print(type(x)) → --> Prints data type → <class 'float'>
```

1.8.1 int: Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example:

```
x = 100
y = 35656222
z = -3255522
print(type(x)) # output → <class 'int'>
print(type(y)) # output → <class 'int'>
print(type(z)) # output → <class 'int'>
```

Representation of integral values in Python:

1. Decimal → Default number system
2. To represent a number in Binary format → Prefix the number with → 0b or 0B
3. To represent a number in Octal format → Prefix the number with → 0o or 0O
4. To represent a number in Hexadecimal format → Prefix the number with → 0x or 0X

Refer Example: int_values_details.py

Base Conversion:

If we print the integral values, output always comes in Decimal format. If we want to print the value explicitly in any number format, we should go with following functions:

1. bin()
2. oct()
3. hex()

Example: base_conversion.py

1.8.2 float :Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Example:

```
x = 1.1011
y = -2.0222
print(type(x)) # output → <class 'float'>
print(type(y)) # output → <class 'float'>
```

1.8.3 complex: Complex numbers are written in the form of a+bj with a "j" as the imaginary part.

Example:

```
x = 13+15j
y = -15j
print(type(x)) # output → <class 'complex'>
print(type(y)) # output → <class 'complex'>
```

1.8.4 str: Strings in python are surrounded by either single quotation marks, or double quotation marks. 'hello' is the same as "hello".

You can display a string literal with the print() function.

```
print("Hello") # output → Hello
print('Hello') # output → Hello
```

1.8.5 bool: Booleans represent one of two values: True or False.

Example: a = True
 b = False

1.8.6 Type Casting: Type Casting is conversion of one data type to another. There are inbuilt functions available to do so.

1. `int()` → Example: `type_cast_to_int.py`
 - float to int → Decimal points will be removed
 - complex to int → Cannot be converted
 - bool to int → Can be
 - str to int → Must be specified in base 10 only, then they can be converted
2. `float()` → Example: `type_cast_to_float.py`
 - int to float → From any int base format (decimal, binary, octal, hexadecimal) to float → possible
 - complex to float → Cannot be converted
 - bool to float → Yes
 - str to float → Must be specified in base 10 only, can contain int or float
3. `complex()` → Example: `type_cast_to_complex.py`
 - 2 functions available to convert any data type to complex format:
 - `complex(arg)`
 - `complex(arg1, arg2)`
 - int to complex → Yes
 - float to complex → Yes
 - bool to complex → Yes
 - str to complex → Yes
4. `bool()` → Example: `type_cast_to_bool.py`
 - zero = false
 - non zero = true
 - empty string = false
 - non empty string = true
 - int to bool → Yes
 - float to bool → Yes
 - complex to bool → Yes
 - str to bool → Yes
5. `str()` → Example: `type_cast_to_str.py`
 - int to str → Yes
 - float to str → Yes
 - complex to str → Yes
 - bool to str → Yes

1.9 Immutability:

Everything in python is an object. If we create an object, we cannot perform any change on it. If we do change, a new object will be created. This non changeable behavior is called as immutability.

Example: `immutability_ex.py`

```
a = 10 → a is the variable , 10 is the object
b = 10
print(id(a))
print(id(b))
print(a is b)
print(a == b)
```

Refer example for more clarification.

1.10**list:**

- Example: list-ex.py
- Lists are used to store multiple items in a single variable.
- Represents group of elements as a single entity.
- Duplicates are allowed.
- Insertion order is preserved.
- Represented by [] square bracket.
- Heterogeneous objects allowed.
- Starts from 0th element.
- We can add/remove elements, so list is dynamic.
- Indexing / slicing concept applicable.
- List is mutable – We can change the content of any index.
- List is grow able in nature.

1.11**tuple:**

- Example: tuple-ex.py
- We cannot add/modify elements to tuple.
- Duplicates are allowed.
- Insertion order is preserved.
- Represented by ()
- Heterogeneous objects allowed.
- Starts from 0th element.
- Indexing / Slicing concept applicable.
- tuple is immutable.
- It is the read only version of list.

1.12**set:**

- Example: set-ex.py
- Ordering is not applicable.
- Ex: s1 = {1,3,4}, to create empty set: s = set()
- Duplicate records not allowed.
- Indexing / slicing not applicable.
- Heterogeneous objects allowed.
- Add new element to set → s.add(10)
- Remove element from set → s.remove(10)
- Set is grow able.
- Set is mutable.

1.13**frozenset:**

- Example: frozenset-ex.py
- It is immutable
- Duplicates not allowed
- Ordering not preserved
- Indexing / slicing not applicable
- Heterogeneous objects allowed
- Add / Remove functionality cannot be done.
- fs = frozenset(set-object)

1.14 range:

- Example: range-ex.py
- To represent a sequence of numbers
- `r = range(n)`
- `r = range(begin, end)`
- `r = range(begin, end, increment/decrement)`
- Indexing/slicing applicable
- Immutable in nature

1.15 dict:

- Example: dict-ex.py
- To store data in key, value pair
- Duplicate keys not allowed
- Values can be duplicated
- Heterogeneous objects allowed
- Mutable in nature
- Indexing/Slicing not applicable
- Empty dict : `d = {}`
- `d = {101:'Ram', 102:'Hari'}`
- `d[101] = 'Sita'`

1.16 bytes, bytearray:

- Example: bytearray-ex.py
- To represent a group of byte values
- Ex: images, audio, video
- Can only represent from 0 to 255
- Bytes is immutable
- Bytearray is mutable

1.17 None Keyword:

- None is used to define a null value or Null object in Python.
- It is not the same as an empty string, a False, or a zero.
- It is a data type of the class `NoneType` object.
- There is no null in Python, we can use None instead of using null values.

Example: `x = None`
`print(x)` `# None`

1.18 Escape Characters :

- Example: esc_char_ex.py
- To insert characters that are illegal in a string, use an escape character.
- An escape character is a backslash `\` followed by the character you want to insert.

1.19 Comments:

- Comments starts with a `#`, and Python will ignore them.
- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.

- Comments can be used to prevent execution when testing code.

Example:

```
#This is a comment  
print("Hello, World!")
```

1.20 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.