

UNIT 4

STATISTICS AND ANALYTICS

20SC02P



INTRODUCTION TO PYTHON:

What is Programming?

Programming is a way to “instruct the computer to perform various tasks”.

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. He is fan of ‘Monty Python’s Flying Circus’, this is a famous TV show in Netherlands, so named after Monty Python.

Python is an open-source, object-oriented and a high-level programming language for web and app development. Also, it is used for software development, system scripting.

Mathematics.

Python over other programming languages:

Python has witnessed incredible growth from the past few years, mainly due to its simple learning, versatility, and efficiency. It is a general-purpose language, designed specifically to be simple to read and write. Furthermore, the language has become a universal platform for various development requirements, and it offers a lot of options to the programmers in general. Other than its easy to use and learn functionality, it offers a mature and supportive python community, which guides developers in the right direction.

Features of Python:

- **Easy to code:** Python is a high-level programming language as it is easy to comprehend as compared to other language like c, c#, Java script, Java and so forth, one can effortlessly learn and code in python barely in hours. Additionally, it is also a developer-friendly language.
- **Platform Independent:** Python programs can be developed and executed on numerous operating system framework. Python can be used on Linux, Windows, Macintosh, Solaris and some others.
- **Object-Oriented Language:** Python bolsters object-oriented language and concepts of classes, objects encapsulation etc.
- **Free and Open Source:** Python language is freely available at the official website. Since, it is open-source, available to the public. So one can download it, use it as well as share it.
- **GUI Programming Support:** Graphical Users interfaces can be made using a module such as PyQt5, PyQt4, wxPython or Tk in python.

- **High-level Language:** Python is a high-level language. When one develops programs in python, he/she didn't need to memorize the system architecture or to manage the memory.
- **Portable language:** Python is a portable language, for instance, on the off chance that the code written in python for windows can also run on different other platforms such as Linux, Unix and Mac etc.
- **Integrated and Interpreted Language:** Python is an Interpreted Language, since python code is executed line by line at a time. Python is additionally an Integrated language since one can without much of a stretch, can integrate python with another language like [C](#), [C++](#) and so on.

How to Download and Install Python 3.9 on Windows

1. Step 1: Download Python 3.9. To start, go to python.org/downloads and then click on the button to download the latest version of Python:
2. Step 2: Run the .exe file. Next, run the .exe file that you just downloaded:
3. Step 3: Install Python 3.9. You can now start the installation of Python by clicking on Install Now:

Syntax of Python:

The syntax of the Python programming language is the set of rules that defines how a Python program will be written and interpreted (by both the runtime system and by human readers). As we know, Python syntax can be executed by writing directly in the command Line:

```
>>> print("Hello, hii!")  
Hello, hii!
```

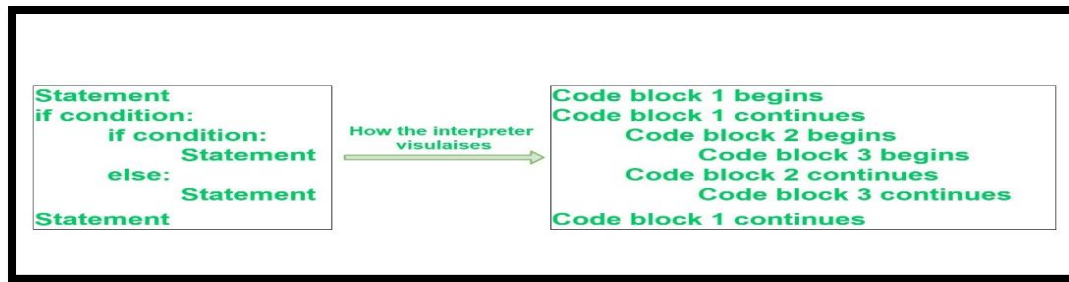
Python Indentation:

Python indentation is something that is a foundation concept for any new python programming, understanding how indentation works is the first step before you can start writing code in python. Indentation refers to the spaces at the beginning of a code line. Python uses indentation to indicate a block of code.

In Python, code block refers to a collection of code that is in the same block or indent. This is most commonly found in classes, functions, and loops.

Python code blocks are useful because they tell you what pieces of code will run as a part of a loop, class, or function.

Indentation in other languages like c, c++, etc is just for readability but in python, the indentation is a very important and mandatory concept that should be followed when writing a python code, otherwise, Indentation Error is thrown by the python interpreter.



Python Indentation Rules:

- Python uses 4 spaces as default indentation spaces. However, the number of spaces can be anything, it is up to the user. But a minimum of one space is needed to indent a statement.
- The first line of python code cannot have Indentation.
- Indentation is mandatory in python to define the blocks of statements.
- You have to use the same number of spaces in the same block of code, otherwise Python will give you an error.
- It is preferred to use whitespaces instead of tabs to indent in python. Also, either use whitespace or tabs to indent, intermixing of tabs and whitespaces in indentation can cause wrong indentation errors.

For more details refer:

<https://www.geeksforgeeks.org/indentation-in-python/>

<https://www.scaler.com/topics/indentation-in-python/>

Example:

```
if 7 > 3:
```

```
    print("Seven is greater than Three!")
```

```
Seven is greater than three!
```

- Python will give you an error if you skip the indentation:

Example

Syntax Error:

```
if 7 > 3:
```

```
    print("Seven is greater than Three!")
```

```
py_compile.PyCompileError: Sorry: IndentationError: expected an indented l
```

Example

Syntax Error:

```
if 7 > 3:
```

```
    print("Seven is greater than three!")
```

```
        print("Seven is greater than three!")
```

```
CompileError: Sorry: IndentationError: unexpected indent (prog.py, line 3)
```

Comments of Python:

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.

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

Example:

```
print ("This is true.")  
# This is false
```

When you run the above code, you will only see the output **This is true**. Everything else is ignored.

Multiline Comments:

- Python does not really have a syntax for multi-line comments.
- To add a multiline comment, you could insert a **#** for each line:

Example :

```
#You cannot use  
#multiple comments  
#in python  
print("You can use in java")
```

The output is:

```
You can use in java
```

Note: Each line that starts with a hash mark will be ignored by the program. Another thing you can do is use multiline strings by wrapping your comment inside a set of triple quotes:

Example:

```
"""You cannot use  
multiple comments  
in python"""  
print("You can use in java")
```

Output:

```
You can use in java
```

Python keywords:

Python keywords are special reserved words that have specific meanings and purposes and can't be used for anything but those specific purposes. These keywords are always available—you'll never have to import them into your code.

Python keywords are different from Python's built-in functions and types. The built-in functions and types are also always available, but they aren't as restrictive as the keywords in their usage.

Keywords in Python				
All keywords has special meaning and purpose in python. Here is list of all keywords in python:				
False	None	True	and	as
assert	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		

For more details refer-<https://realpython.com/python-keywords/>

Python built-in functions:

The Python built-in functions are defined as the functions whose functionality is pre-defined in Python. The python interpreter has several functions that are always present for use. These functions are known as Built-in Functions. There are several built-in functions in Python which are listed below:

Built-in Functions			
A <code>abs()</code> <code>all()</code> <code>any()</code> <code>ascii()</code>	E <code>enumerate()</code> <code>eval()</code> <code>exec()</code>	L <code>len()</code> <code>list()</code> <code>locals()</code>	R <code>range()</code> <code>repr()</code> <code>reversed()</code> <code>round()</code>
B <code>bin()</code> <code>bool()</code> <code>breakpoint()</code> <code>bytearray()</code> <code>bytes()</code>	F <code>filter()</code> <code>float()</code> <code>format()</code> <code>frozenset()</code>	M <code>map()</code> <code>max()</code> <code>memoryview()</code> <code>min()</code>	S <code>set()</code> <code>setattr()</code> <code>slice()</code> <code>sorted()</code> <code>staticmethod()</code>
C <code>callable()</code> <code>chr()</code> <code>classmethod()</code> <code>compile()</code> <code>complex()</code>	G <code>getattr()</code> <code>globals()</code>	N <code>next()</code>	<code>str()</code> <code>sum()</code> <code>super()</code>
D <code>delattr()</code> <code>dict()</code> <code>dir()</code> <code>divmod()</code>	H <code>hasattr()</code> <code>hash()</code> <code>help()</code> <code>hex()</code>	O <code>object()</code> <code>oct()</code> <code>open()</code> <code>ord()</code>	T <code>tuple()</code> <code>type()</code>
	I <code>id()</code> <code>input()</code> <code>int()</code> <code>isinstance()</code> <code>issubclass()</code> <code>iter()</code>	P <code>pow()</code> <code>print()</code> <code>property()</code>	V <code>vars()</code>
			Z <code>zip()</code>
			_ <code>__import__()</code>

Data types of Python:

- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.
- Python has the following data types built-in by default, in these categories:

Text Type:	Str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey": "value", "name": "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

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

Example 1:

Print the data type of the variable x:

```
x = 20

#display x:
print(x)

#display the data type of x:
print(type(x))
```

Output:

```
20
<class 'int'>
```

Example 2 :

```
x = "Hello World"

#display x:
print(x)

#display the data type of x:
print(type(x))
```

Output:

```
Hello World
<class 'str'>
```

Variables in python:

A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

Python Variable Types:

Every value in Python has a datatype. Different data types in Python are Numbers, List, Tuple, Strings, Dictionary, etc. Variables in Python can be declared by any name or even alphabets like a, aa, abc, etc.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total volume).

Rules for creating variables in Python:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case-sensitive (name, Name and NAME are three different variables).
- The reserved words(keywords) cannot be used naming the variable.

Example:

Legal Variable Names are.

```
myvar = "Python"
my_var = "Python"
_my_var = "Python"
myVar = "John"
MYVAR = "John"
myvar2 = "John"

print(myvar)
print(my_var)
print(_my_var)
print(myVar)
print(MYVAR)
print(myvar2)
```

Output:

```
Python
Python
Python
John
John
John
```

- Python allows you to assign values to multiple variables in one line:

Example:

```
x, y, z = "Mangoe", "Banana", "Apple"

print(x)
print(y)
print(z)
```

OutPut:

```
Mangoe
Banana
Apple
```

- Python allows you to assign the *same* value to multiple variables in one line:

Exampe :

```
a = b = c = "Apple"

print(a)
print(b)
print(c)
```

OutPut :

```
Apple
Apple
Apple
```

Output Variables:

- The Python **print** statement is often used to output variables.
- To combine both text and a variable, Python uses the **+** character.
- You can also use the **+** character to add a variable to another variable.
- For numbers, the **+** character works as a mathematical operator.

User Input

- Python allows for user input.
- The method is a bit different in Python 3.6 than Python 2.7.
- Python 3.6 uses the **input()** method.

Python operators:

Python operator is a symbol that performs an operation on one or more operands. An operand is a variable or a value on which we perform the operation.

Python Operator falls into 7 categories:

- Python Arithmetic Operator
- Python Relational Operator(comparison)
- Python Assignment Operator
- Python Logical Operator
- Python Membership Operator
- Python Identity Operator
- Python Bitwise Operator

Python Arithmetic Operators:

Operator	Name	Description	Syntax	Example
+	Addition	Performs addition	$c = a + b$	$a = 5, b = 5$ then $c = 10$
-	Subtraction	Performs subtraction	$c = a - b$	$a = 5, b = 3$ then $c = 2$
*	Multiplication	Performs multiplication	$c = a * b$	$a = 5, b = 5$ then $c = 25$
/	Division	Performs division	$c = a / b$	$a = 10, b = 5$ then $c = 2$
%	Modulus	Performs division but returns the remainder	$c = a \% b$	$a = 15, b = 2$ then $c = 1$
//	Floor Division	Performs division but returns the quotient in which the digits after the decimal points are removed	$c = a // b$	$a = 15, b = 2$ then $c = 7$
**	Exponent	Performs multiplication to power raised	$c = a ** b$	$a = 2, b = 4$ then $c = 16$

Python Comparison Operator:

Python Comparison Operators		
Operator	Name	Example
=	Equal	$a = b$
!=	Not equal	$a != b$
>	Greater than	$a > b$
<	Less than	$a < b$
>=	Greater than or equal to	$a >= b$
<=	Less than or equal to	$a <= b$

Python Assignment Operator:

Operator	Example	Equivalent Expression
=	$m = 10$	$m = 10$
+=	$m += 10$	$m = m + 10$
-=	$m -= 10$	$m = m - 10$
*=	$m *= 10$	$m = m * 10$
/=	$m /=$	$m = m / 10$
%=	$m \% = 10$	$m = m \% 10$
<<=	$a <<= b$	$a = a << b$
>>=	$a >>= b$	$a = a >> b$
>>>=	$a >>>= b$	$a = a >>> b$
&=	$a \&= b$	$a = a \& b$
^=	$a \wedge= b$	$a = a \wedge b$
=	$a = b$	$a = a b$

Python Logical Operator

OPERATION	RESULT	NOTES
X or Y	If X is False, then Y, else X	Y is executed only if X is False Else if X is true, X is result.
X and Y	If X is false, then X else Y	Y is executed only if X is true, else if X is false , X is result.
not X	if X is true, then false, else true	not has lower priority than non - boolean operators. Eg. not a==b ==> not (a==b)

Python Membership Operator

Operator	Meaning	Example
In	True if value/variable found in the sequence	5 in x
Not in	True if value/variable is not found in sequence	5 not in x

Python Identity Operator

Identity Operators are used to check the address reference of two variable is same or not.

Operator	Description
is	It is evaluated to be true if the reference present at both sides points to the same object. a = 10 , b = 10 , a is b returns true.
is not	It is evaluated to be true if the reference present at both side do not point to the same object. a = 10 , b = 20 a is not b returns true

Python Bitwise Operator:

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(a & b) will give 12 which is 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(a b) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(a ^ b) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~a) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	a << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 will give 15 which is 0000 1111

String Concatenation:

String Concatenation (ADDITION) is a very common operation in programming. Python String Concatenation can be done using various ways. This session is aimed to explore different ways to concatenate strings in a python program.

We can perform string concatenation in the following methods:

- Using + operator
- Using join () method
- Using % operator
- Using format () function

String concatenation using (+) operator

This is the simplest way of string concatenation. Let's look at a simple example.

```
s1 = 'mallika'
s2 = 'gpt'
s3 = 'bantwal'
s4 = s1 + s2 + s3
print(s4)
```

Output: mallikagptbantwal

If-else in Python:

While writing code in any language, you will have to control the flow of your program. This is generally the case when there is decision making involved - you will want to execute a certain line of codes if a condition is satisfied, and a different set of code in case it is not. In Python, you have the if, elif and the else statements for this purpose.

Simple if statement

This is the simplest example of a conditional statement. The syntax is:

```
if(condition):
    indented Statement Block
```

The block of lines indented the same amount after the colon (:) will be executed whenever the condition is TRUE.

Single test: if-else Statement

The if-else statement is used to code the same way you would use it in the English language. The syntax for the if-else statement is:

```
if(condition):  
    Indented statement block for when condition is TRUE  
else:  
    Indented statement block for when condition is FALSE
```

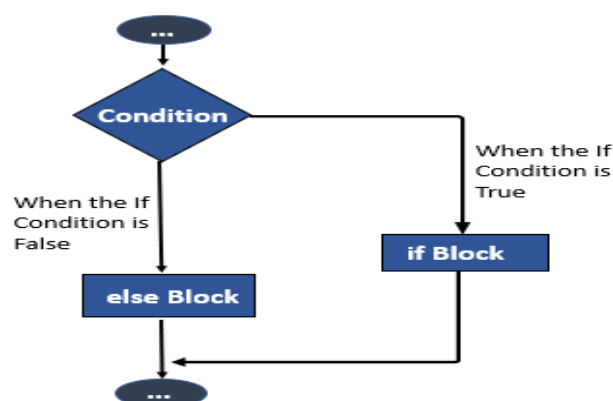
The syntax followed by the if-else-if statement is:

```
if(Condition1):  
    Indented statement block for Condition1  
elif(Condition2):  
    Indented statement block for Condition2  
else:  
    Alternate statement block if all condition check above fails.
```

➤ The syntax of the *if...else* statement is

```
if expression:  
    statement(s)  
else:  
    statement(s)
```

- The if..else statement evaluates test expression and will execute the body of if only when the test condition is True.
- If the condition is False, the body of else is executed. Indentation is used to separate the blocks.



Loops in Python.

THE WHILE LOOP:

With the while loop we can execute a set of statements as long as a condition is true.

Print i as long as i is less than 8:

```
i = 1
```

```
while i < 8:
```

```
print(i)
i += 1
```

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

THE BREAK STATEMENT

With the break statement we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

THE CONTINUE STATEMENT

With the continue statement we can stop the current iteration, and continue with the next:

Continue to next iteration if i is 5

```
i = 0
while i < 10:
    i += 1
    if i == 5:
        continue
    print(i)
```

THE ELSE STATEMENT

With the else statement we can run a block of code once when the condition no longer is true:

Print the message once the condition is false

```
i = 1
while i < 4:
    print(i)
    i += 1
else:
    print("not satisfied")
```

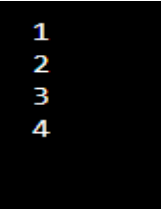
Python has two primitive loop commands:

- **while** loops
- **for** loops
 - With the **while** loop we can execute a set of statements as long as a condition is true.

Example: print i as long as i is less than 5

```
i = 1
while i < 5:
    print(i)
    i = i+1
```

OutPut:



```
1
2
3
4
```

To write the programme for finding sum of first 10 natural numbers, follow the following steps:

Step 1: User must enter the number of natural numbers to find the sum of.

Step 2. The sum variable is initialized to 0.

Step 3. The while loop is used to find the sum of natural numbers and the number is decremented for each iteration.

Step 4. The numbers are added to the sum variable and this continues till the value of the variable is greater than 0.

Step 5. When the value of the variable becomes lesser than 0, the total sum of N natural numbers is printed.

PYTHON LIBRARIES:

Normally, a library is a collection of books or is a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, and values, etc.

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc.

❖ **Matplotlib**

Matplotlib is a Python library that uses Python Script to write 2-dimensional graphs and plots. Often mathematical or scientific applications require more than single axes in a representation. This library helps us to build multiple plots at a time. However, Matplotlib is used to manipulate different characteristics of figures as well.

❖ **Numpy**

Numpy is a popular array – processing package of Python. It provides good support for different dimensional array objects as well as for matrices. Numpy is not only

confined to providing arrays only, but it also provides a variety of tools to manage these arrays. It is fast, efficient, and really good for managing matrices and arrays.

❖ **Scipy**

Scipy is an open-source python library that is used for both scientific and technical computation. It is a free python library. And very suitable for machine learning. However, computation is not the only task that makes scipy special. It is also very popular for image manipulation, as well.

❖ **Pandas**

Pandas is a python software package. It is a must to learn for data-science and dedicatedly written for Python language. It is a fast, demonstrative, and adjustable platform that offers intuitive data-structures. You can easily manipulate any type of data such as – structured or time-series data with this amazing package.

❖ **xlrd**

xlrd is a library for reading data and formatting information from Excel files in the historical .xlsx format. This library will no longer read anything other than .xlsx files. For alternatives that read newer file formats, please see <http://www.python-excel.org/>.

❖ **Openpyxl**

openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files.

It was born from lack of existing library to read/write natively from Python the Office Open XML format.

Before the variables such as mean, mode, median and standard deviation of the data is found, one has to learn how to import an excel file into python environment.

Pandas is the library which supports for the analysis of the data

The following is the simple step involved in importing the file

```
import pandas as pd
df=pd.read_excel("C:\\Users\\Jogi\\Desktop\\bar graph.xlsx")
print(df)
```

Here df is nothing but the data frame.

Path of the file has to be typed correctly with double backward slash.

Python Numpy:

- If you have [Python](#) and [PIP](#) already installed on a system, then installation of NumPy is very easy.

- Install it using this command:

```
C:\Users\Your Name>pip install numpy
```

- Once NumPy is installed, import it in your applications by adding the **import** keyword:
`import numpy`
- NumPy is usually imported under the **np** alias.

For more information regarding libraries refer:

<https://www.ubuntupit.com/best-python-libraries-and-packages-for-beginners/>

Python Functions and Arrays:

In Python, a function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- In Python a function is defined using the **def** keyword.
- To call a function, use the function name followed by parenthesis.
- An array is a special variable, which can hold more than one value at a time.
- Use the **len()** method to return the length of an array (the number of elements in an array).

CREATING A FUNCTION:

In Python a function is defined using the keyword: **def**

```
def_function():  
    print("namaste from a function")
```

CALLING A FUNCTION.

To call a function, use the function name followed by parenthesis:

```
def my_function():  
    print("Hello from a function")  
my_function()
```

PASSING A LIST AS AN ARGUMENT:

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

```
def my_function(food):  
    for x in food:  
        print(x)  
  
fruits = ["CHAPATHI", "IDLY", "DHUMBIRIYANI"]  
  
my_function(INDIAN FOOD)
```

RETURN VALUES:

To let a function return a value, use the **return** statement:

```
def my_function(x):  
    return 5 * x  
  
print(my_function(3))  
print(my_function(5))  
print(my_function(9))
```

THE PASS STATEMENT:

Function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

```
def myfunction():  
    pass
```

RECURSION:

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, **tri_recursion()** is a function that we have defined to call itself ("recurse"). We use the **k** variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

```
def tri_recursion(k):
    if(k > 0):
        result = k + tri_recursion(k - 1)
        print(result)
    else:
        result = 0
    return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```