INTRODUCTION TO PYTHON PROGRAMMING LANGUAGE.

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

- •Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- •Python can be treated in a procedural way, an object-orientated way or a functional way.

Syntax compared to other programming languages

- •P[1]ython was designed for readability, and has some similarities to the English language with influence from mathematics.
- •Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- •Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
print("Hello, World!")
```

Congratulations, you have written your first Python program.

1 mbaabuharun8@gmail.com
#0798166628

Python Comments

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.

Creating a Comment

Comments starts with a #, and Python will ignore them:

```
#this is a comment
print(“hello world”)
```

output will be hello world.

Comments can also be placed at the end of a line, and Python will ignore the rest of the line:

```
print(“hello python”) #this outputs  hello python
```

Comments does not have to be text to explain the code, it can also be used to prevent Python from executing code.

```
#adding two numbers
#print (sum of a and b)
a=20
b=30
print(a+b)
#output 50
```

Multi Line Comments

Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line, like the above example.

Also, since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place you comment inside it:

```
“””
THIS IS FIRST LINE COMMENT
THE SECOND LINE COMMENT
THIS IS A NOTHER COMMENT
“”””
print(“hello world”)
```

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

PythonVariables.

Variables are containers for storing data values.

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

```
X=20
y=40
print(x)
print(y)
```

Variables do not need to be declared with any particular type and can even change type after they have been set.

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- •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 (age, Age and AGE are three different variables)

Remember that variable names are case-sensitive

Python Numbers

There are three numeric types in Python:

- •int
- •float
- •complex

Variables of numeric types are created when you assign a value to them:

```
x =1# int
y =2.8# float
z = 1j# complex
```

To verify the type of any object in Python, use the `type()` function:

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

Int

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

```
x =1
y =35656222554887711
z =-3255522

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

Type Conversion

You can convert from one type to another with the `int()` `float()`, and `complex()` methods:

Example

Convert from one type to another

```
#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

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

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

Note: You cannot convert complex numbers into another number type.

Random Number

Python does not have a `random()` function to make a random number, but Python has a built-in module called `random` that can be used to make random numbers:

Example

Import the random module, and display a random number between 1 and 9:

```
import random

print(random.randrange(1,10))
```

Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

- `int()` - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
- `float()`- constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- `str()`- constructs a string from a wide variety of data types, including strings, integer literals and float literals

Example

Integers:

```
x =  int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Example

Floats:

```
x =  float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
```

```
z =  float("3")    # z will be 3.0
w = float("4.2") # w will be 4.2
```

Strings:

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Python Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Python Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	**Description**	**Example**
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	**Description**	**Example**
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

---

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	**Description**	**Example**
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

---

Exercise:

Multiply `10` with `5`, and print the result.

```
print(10 * 5)
```

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.