Python

- Python is a high-level object-oriented programming language.
- It is a general-purpose programming language that can be used for programming, system scripting, mathematics, web development & software development unlike other languages like HTML & java.
- Python was developed in the early 1990’s by Guido van Rossum.
- In February 1991, Guido Van Rossum published the code (labeled version 0.9.0) to alt. sources.
- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.

Python Features

- Python provides many useful features which make it popular and valuable from the other programming languages.
- It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation.

❖Few essential features are listed below:

1. **Easy to Learn and Use** :
   - Python is easy to learn as compared to other programming languages.
   - Its syntax is straightforward and much the same as the English language.
   - There is no use of the semicolon or curly-bracket, the indentation defines the code block.
2. **Expressive Language:**
   - Python can perform complex tasks using a few lines of code.
   - A simple example, the hello world program you simply type print("Hello World").
   - It will take only one line to execute, while Java or C takes multiple lines.

3. Interpreted Language:

- Python is an interpreted language; it means the Python program is executed one line at a time.
- The advantage of being interpreted language, it makes debugging easy and portable.

4. Cross-platform Language:

- Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc.
- So, we can say that Python is a portable language.
- It enables programmers to develop the software for several competing platforms by writing a program only once.

5. Free and Open Source:

- Python is freely available for everyone.
- It has a large community across the world that is dedicatedly working towards make new python modules and functions.
- Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6. Object-Oriented Language:

- Python supports object-oriented language and concepts of classes and objects come into existence.
- It supports inheritance, polymorphism, and encapsulation, etc.
- The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7. Extensible :

- It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code.
- It converts the program into byte code, and any platform can use that byte code.

8. Large Standard Library :

- It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting.
- There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc.

9. GUI Programming Support:

- Graphical User Interface is used for the developing Desktop application.
- PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

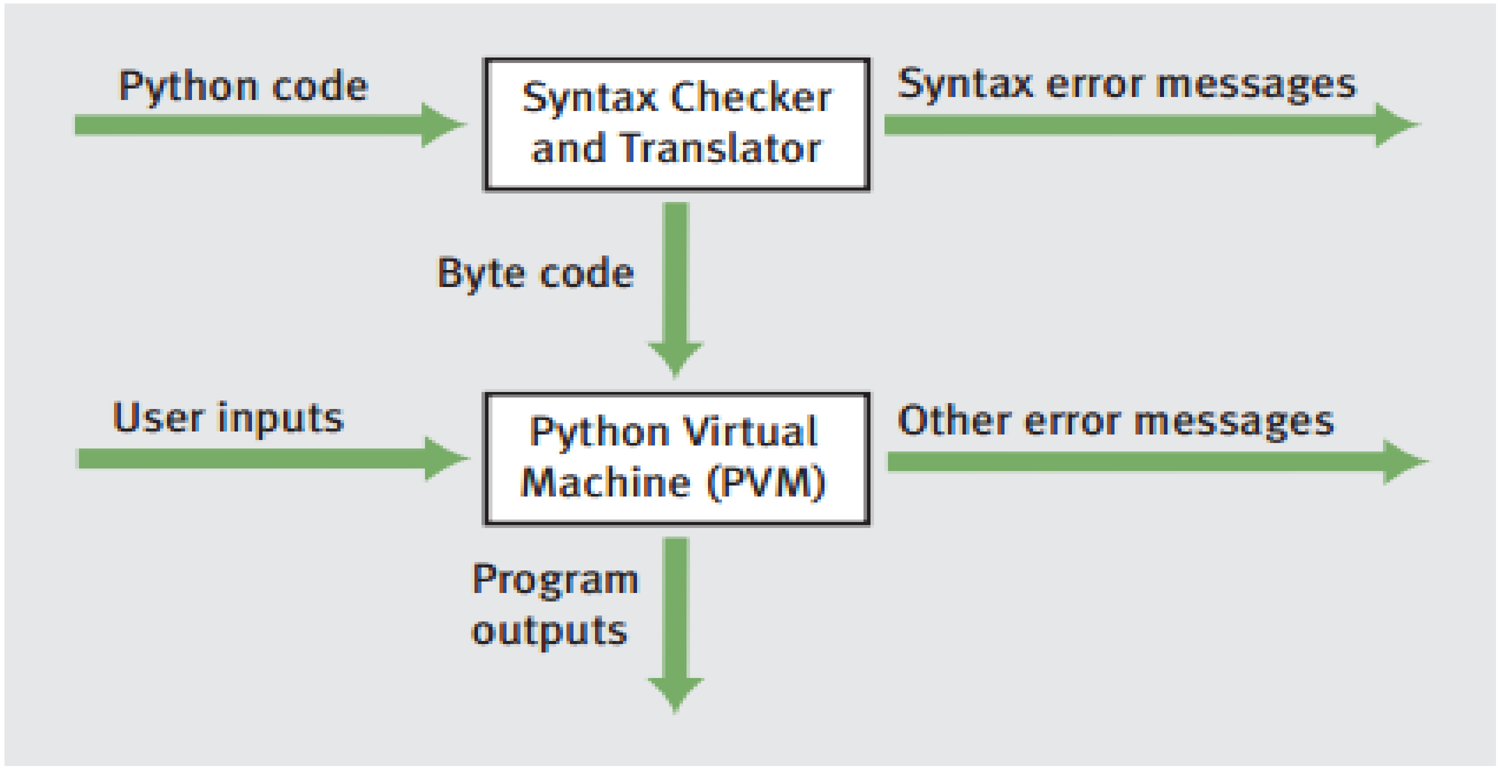
10. Integrated:

- It can be easily integrated with languages like C, C++, and JAVA, etc.
- Python runs code line by line like C,C++ Java. It makes easy to debug the code.

11. Dynamic Memory Allocation :

- In Python, we don't need to specify the data-type of the variable.
- When we assign some value to the variable, it automatically allocates the memory to the variable at run time.
- Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

Viewing Of Byte Code



- Python is usually called an interpreted language, however, it combines compiling and interpreting.

- When we execute a source code (a file with a .py extension), Python first compiles it into a bytecode.
- The bytecode is a low-level platform-independent representation of your source code, however, it is not the binary machine code and cannot be run by the target machine directly.
- In fact, it is a set of instructions for a virtual machine which is called the Python Virtual Machine (PVM).
- After compilation, the bytecode is sent for execution to the PVM.
- The PVM is an interpreter that runs the bytecode and is part of the Python system. The bytecode is platform-independent, but PVM is specific to the target machine.
- The default implementation of the Python programming language is CPython which is written in the C programming language. CPython compiles the python source code into the bytecode, and this bytecode is then executed by the CPython virtual machine.

Flavors of Python:

- **CPython**: It is the standard flavor of Python. It can be used to work with C language Applications.
- **Jython OR JPython:** It is for Java Applications. It can run on JVM
- **IronPython:** It is for C#.Net platform
- **PyPy:** The main advantage of PyPy is performance will be improved because JIT compiler is available inside PVM.
- **RubyPython :** For Ruby Platforms
- **AnacondaPython :** It is specially designed for handling large volume of data processing.

Memory Management In Python

- Memory allocation can be defined as allocating a block of space in the computer memory to a program.

- **Static Memory Allocation :**
  - Static memory allocation happens at the compile time .
  - For example - In C/C++, we declare a static array with the fixed sizes.
  - Memory is allocated at the time of compilation.

- **Dynamic Memory Allocation**:
  - Dynamic memory allocates the memory at the runtime to the program.
  - For example - In C/C++, there is a predefined size of the integer of float data type but there is no predefine size of the data types.

Memory Management In Python

  - Memory is allocated to the objects at the run time. We use the Heap for implement dynamic memory management. We can use the memory throughout the program.
  - Everything in Python is an object means dynamic memory allocation inspires the Python memory management.
  - Python memory manager automatically vanishes when the object is no longer in use.
- In Python memory allocation and deallocation method is automatic as the Python developers created a garbage collector for Python so that the user does not have to do manual garbage collection.

Comparisons between c-Java-Python

Sr. No.	Parameters	C	Java	Python
1	Language Type	Procedure oriented programming Language	Object oriented programming Language	Both type of programming Language
2	Language level	Middle level Language	High level Language	High level Language
3	Building block	Function driven	Object and class driven	Function ,class and object driven
4	Extensions	.c	.java	.py
5	Platform	Dependent	Independent	Independent
6	Comment style	/*-------*/	// for single line /*-------*/for multiline	# for single line ‘’‘ ------’“ for multiline
7	keywords	32	63	33
8	Data security	Not secured	Fully secured	Secured(less than java)

- Hello World Program:
  - In Java :

```
class HelloWorld
{
        public static void main(String args[])
             {
                  System.out.println("Hello World");
             }
}
```

  - In C:

```
void  main()
{
                printf("Hello World");
}
```

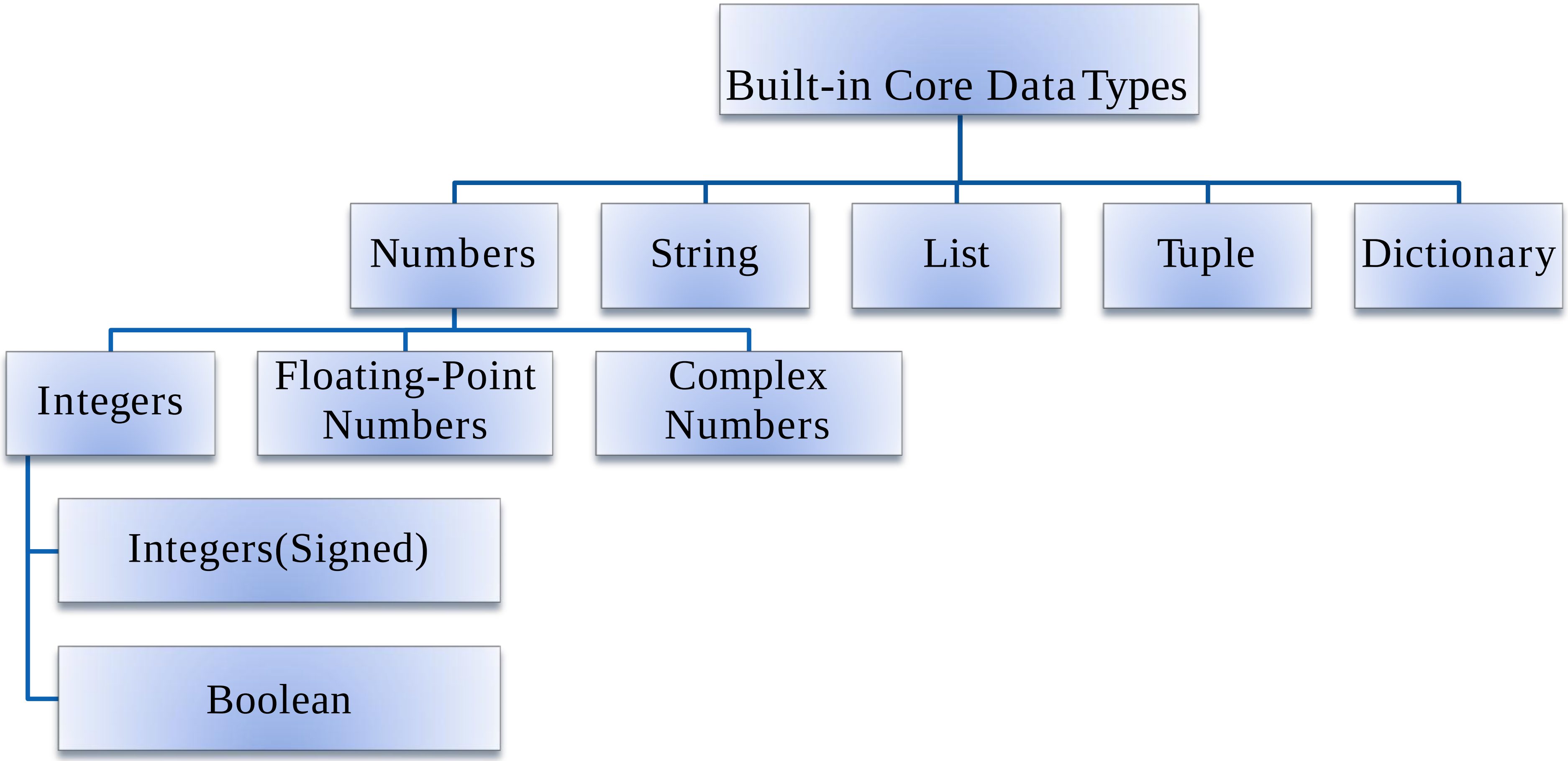
  - In Python :

```
print("Hello World")
```

Python Built-in Core Data Types

Python offers following built-in core data types :

i) Numbers ii) String iii) List iv) Tuple v) Dictionary



Integers

- Integers are whole numbers.
- They have no fractional parts.
- Integers can be positive or negative.

There are two types of integers in Python:

i) **Integers(Signed)** : It is the normal integer representation of whole numbers using the digits 0 to 9. Python provides single int data type to store any integer whether big or small. It is signed representation i.e. it can be positive or negative.

ii) **Boolean :** These represent the truth values True and False. It is a subtype of integers and Boolean values True and False corresponds to values 1 and 0 respectively

Demonstration of Integer Data Type

```
#Demonstration of Integer-Addition of two integer
number   a=45
b=67
sum=a+b
print("The sum of two integers=",sum)
```

Output:

```
The sum of two integers= 112
```

Floating Point Numbers

A number having fractional part is a floating point number. It has a decimal point.

It is written in two forms :

i) Fractional Form : Normal decimal notation e.g. 675.456

ii) Exponent Notation: It has mantissa and exponent. e.g. 6.75456E2

Advantage of Floating point numbers:

They can represent values between the integers. They can represent a much greater range of values.

Disadvantage of Floating point numbers:

Floating-point operations are usually slower than integer operations.

Demonstration of Floating Point Data Type

```
#Demonstration of Float Number- Calculate Simple Interest
princ=float(input("Enter the Principal Amount:"))
rate=float(input("Enter the Rate of interest:"))
time=float(input("Enter the Time period:"))
si=(princ*rate*time)/100
print("The Simple Interest=",si)
```

Output:

```
Enter the Principal Amount:5000
Enter the Rate of interest:8.5
Enter the Time period:5.5
Simple Interest= 2337.5
```

Complex Number

Python represents complex numbers in the form a+bj.

```
#Demonstration of Complex Number- Sum of two Complex Numbers
a=7+8j
b=3.1+6j
c=a+b
print("Sum of two Complex Numbers")
print(a,"+",b,"=",c)
```

Output:

```
(7+8j) + (3.1+6j) = (10.1+14j)
```

Strings

A String is a group of valid characters enclosed in Single or Double quotation marks. A string can group any type of known characters i.e. letters ,numbers and special characters.

A Python string is a sequence of characters and each character can be accessed by its index either by forward indexing or by backward indexing.

e.g. subj="Computer"

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Subj	C	o	m	p	u	t	e	r
	-8	-7	-6	-5	-4	-3	-2	-1

Demonstration of String Data Type

```
#Demonstration of String- To input string & print it
my_name=input("What is your Name? :")
print("Greetings!!!")
print("Hello!",my_name)
print("How do you do?")
```

Output :

```
What is your Name? : Vaishali  Patel
Greetings!!!
Hello! Vaishali  Patel
How do you do?
```

List

- The List is Python’s compound data type.
- A List in Python represents a list of comma separated values of any data type between square brackets.
- Lists are Mutable.

```
#Demonstration of List-1 Program to input 2 list & join it
List1=eval(input("Enter Elements for List 1:"))
List2=eval(input("Enter Elements for List 2:"))
List=List1+List2
print("List 1 :",List1)
print("List 2 :",List2)
print("Joined List :",List)
```

Output:

```
Enter Elements for List 1:[12,78,45,30]
Enter Elements for List 2:[80,50,56,77,95]
List 1 : [12, 78, 45, 30]
List 2 : [80, 50, 56, 77, 95]
Joined List : [12, 78, 45, 30, 80, 50, 56, 77, 95]
```

Tuple

- The Tuple is Python’s compound data type.
- A Tuple in Python represents a list of comma separated values of any data type Within parentheses.
- Tuples are Immutable.

```
#Demonstration of Tuple- Program to input 2 tuple & join it
tuple1=eval(input("Enter Elements for Tuple 1:"))
tuple2=eval(input("Enter Elements for Tuple 2:"))
Tuple=tuple1+tuple2
 print(“Tuple 1 :“,tuple1)
print(“Tuple 2 :“,tuple2)
print("Joined  Tuple :“,Tuple)
```

Output:

```
Enter Elements for Tuple 1:(12,78,45,30)
Enter Elements for Tuple 2:(80,50,56,77,95)
List 1 : (12, 78, 45, 30)
List 2 : (80, 50, 56, 77, 95)
Joined List : (12, 78, 45, 30, 80, 50, 56, 77, 95)
```

Dictionary

- Dictionaries are unordered collection of elements in curly braces in the form of a key:value pairs that associate keys to values.
- Dictionaries are Mutable.
- As dictionary elements does not have index value ,the elements are accessed through the keys defined in key:value pairs.

```
#Demonstration of Dictionary- Program to save Phone nos. in dictionary &
    print it
Phonedict={“Samir”:9876547643,”Ramesh”:7660784507,”Mahesh”:9156736
    540,”Suresh”:8153679840}
print(Phonedict)
```

Output:

```
{' Samir ': 9876547643, ' Ramesh': 7660784507, ' Mahesh': 9156736540,
    ' Suresh': 8153679840}
```

Example:

- **Find the area of a circle given the radius:**

```
Radius = 10
pi = 3.14159
area = pi * Radius * Radius
print( area )
```

- **will print 314.15 to the screen.**

Variables

- Variables are containers for storing data values.
- Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.

Ex:

```
x = 5
y = “Python“
print(x)
print(y)
```

Output:

```
5
Python
```

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

Ex:

```
x = 4
x = “Python"
print(x)
```

Ouput: Python

- You can get the data type of a variable with the type() function.

Ex:

```
x = 5
y = “Python“
type(x)
type(y)
```

Output:

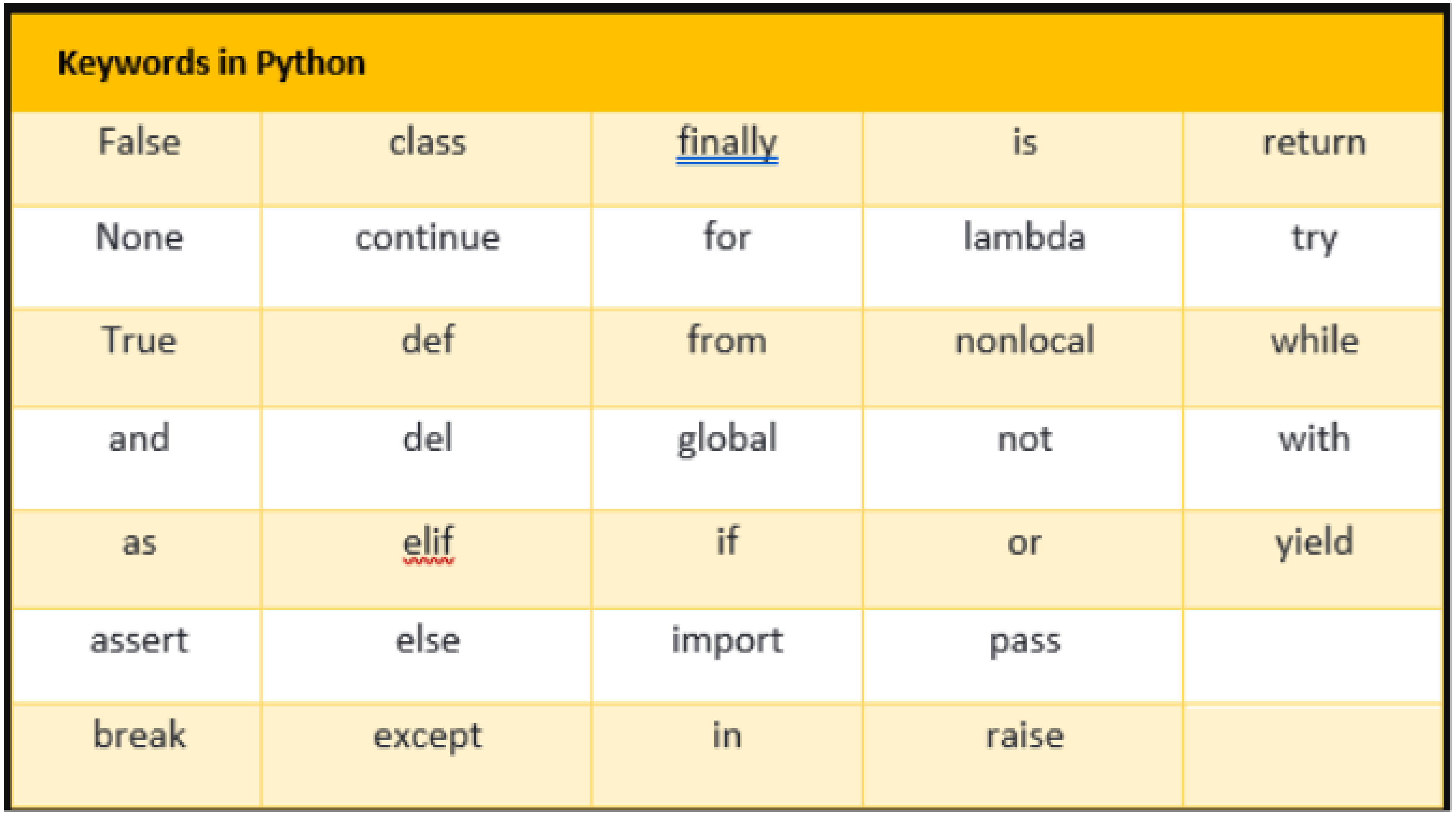
```
<class 'int'>
<class 'str'>
```

Identifiers and reserved Words

Keywords:

- 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.

- The list of all the keywords is given below.



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

Identifiers:

- 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:
  - Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.

  Names like myClass, var_1 and print_this_to_screen, all are valid example.
  - An identifier cannot start with a digit. 1variable is invalid, but variable1 is a valid name.
  - Keywords cannot be used as identifiers.
  - We cannot use special symbols like !, @, #, $, % etc. in our identifier.
  - An identifier can be of any length.

Type Conversion

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.
- Python has two types of type conversion.
  - Implicit Type Conversion
  - Explicit Type Conversion

- **Implicit Type Conversion**:

  In Implicit type conversion, Python automatically converts one data type to another data type.

Example:

```
num_int = 45
num_flo = 7.65
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Output:

```
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>
Value of num_new: 52.65
datatype of num_new: <class 'float'>
```

- **Explicit Type Conversion:**
- In Explicit Type Conversion, users convert the data type of an object to required data type.
- We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion.
- This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.
- **Example:**

```
num_int = 45
num_str = “23"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))
 num_str = int(num_str)
 print("Data type of num_str after Type Casting:",type(num_str)) num_sum
= num_int + num_str
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Output:

```
Data type of num_int: <class 'int'>
 Data type of num_str before Type Casting: <class 'str'>
Data type of num_str after Type Casting: <class 'int'>
Sum of num_int and num_str: 68
Data type of the sum: <class 'int'>
```