Introduction to Python Progrgamming

Python is a very expressive language, which means that we can usually write far fewer lines of Python code than would be required for an equivalent application written in, say, C++ or Java.

Python is a cross-platform language: In general, the same Python program can be run on Windows and Unix-like systems such as Linux, BSD, and Mac OS X, simply by copying the file or files that make up the program to the target machine, with no “building” or compiling necessary.

One of Python’s great strengths is that it comes with a very complete standard library And in addition to the standard library, thousands of third- party libraries are available, some providing more powerful and sophisticat- ed facilities than the standard library—for example, the Twisted networking library and the NumPy numeric library

Python can be used to program in procedural, object-oriented, and to a lesser extent, in functional style, although at heart Python is an object-oriented language.

Creating and Running Python Programs

Python files normally have an extension of .py

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

```
Hello World!
```

Python programs are executed by the Python interpreter

Single Line Comment

In Python, comments begin with a # and continue to the end of the line.

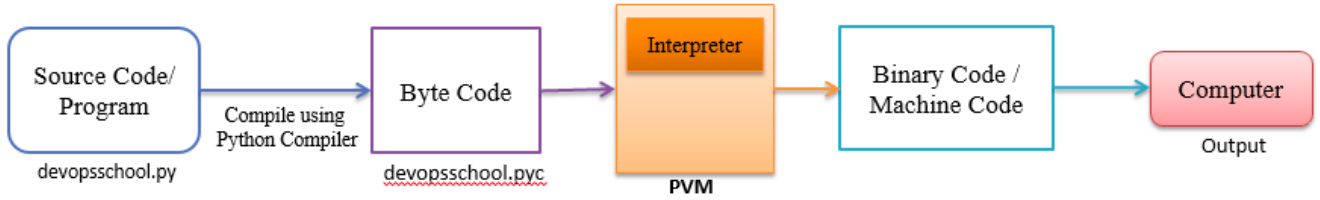
Multi-Line Comment

Triple-quotes are used to comment multiple lines

Python Virtual Machine (PVM)

Python Virtual Machine (PVM) is a program which provides programming environment. The

role of PVM is to convert the byte code instructions into machine code so the computer can execute those machine code instructions and display the output.



Identifiers and Keywords

The names we give to our object references are called identifiers or just plain names.

A valid Python identifier is a nonempty sequence of characters of any length that consists of a “start character” and zero or more “continuation characters”.

The start char- acter can be anything that Unicode considers to be a letter, including the ASCII letters (“a”, “b”, ..., “z”, “A”, “B”, ..., “Z”), the underscore (“_”), as well as the let- ters from most non-English languages.

Each continuation character can be any character that is permitted as a start character, or pretty well any non- whitespace character, including any character that Unicode considers to be a digit, such as (“0”, “1”, ..., “9”)

=> Identifiers are case-sensitive

so for example, TAXRATE , Taxrate , TaxRate , taxRate , and taxrate are five different identifiers.

=>no identifier can have the same name as one of Python’s keywords

Variable

Variables are used to give a name to a memory location that holds a value.

Data Types

One fundamental thing that any programming language must be able to do is represent items of data.

Python provides several built-in data types

Integral Types

Python provides two built-in integral types, int and bool . ★ Both integers and Booleans are immutable =>When used in Boolean expressions, 0 and False are False , and any other integer and True are True . =>When used in numerical expressions True evaluates to 1 and False to 0.

for example, we can increment an integer, i , using the expression i += True .

Floating-Point Types

floating-point values: the built-in float and complex types =>The complex data type is an immutable type that holds a pair of float s, one representing the real part and the other the imaginary part of a complex number.

=>Literal complex numbers are written with the real and imaginary parts joined by a + or - sign, and with the imaginary part followed by a j . ★ Here are some examples: 3.5+2j , 0.5j , 4+0j , -1-3.7j . Notice that if the real part is 0, we can omit it entirely.

Strings

Strings are represented by the immutable str data type which holds a sequence of Unicode characters.

=>we are free to use single or double quotes providing we use the same at both ends

=>In addition, we can use a triple quoted string—this is Python-speak for a string that begins and ends with three quote characters (either three single quotes or three double quotes).

Slicing and Striding Strings

in- dividual characters in a string, can be extracted using the item access operator ([]).

=>In fact, this operator is much more versatile and can be used to extract not just one item or character, but an entire slice (subsequence) of items or charac- ters, in which context it is referred to as the slice operator.

=> The slice operator has three syntaxes:

seq[start]

seq[start:end]

seq[start:end:step]

String Operators and Methods

As strings are sequences they are “sized” objects, and therefore we can call Sized len() with a string as the argument.

=> We have seen that the + operator is overloaded to provide string concatenation. In cases where we want to concatenate lots of strings the str.join() method

=> The str.join() method can also be used with the built-in reversed() function, to reverse a string, for example, "".join(reversed(s)) , although the same result can be achieved more concisely by striding, for example, s[::-1] .

=>The * operator provides string replication:

```
s = "=" * 5
s
```

```
'====='
```

=> str.index()

str.find()

returns the index position of the sub- string, or -1 on failure.

Both methods take the string to find as their first ar- gument, and can accept a couple of optional arguments.

String Formatting

The str.format() method returns a new string with the replacement fields in its string replaced with its arguments suitably formatted

```
"The novel '{0}' was published in {1}".format("Hard Times", 1854)
```

```
'The novel 'Hard Times' was published in 1854'
```

If we try to concatenate a string and a number

```
"{0}{1}".format("The amount due is $", 200)
```

```
'The amount due is $200'
```

Operators

Operators are used to perform operations on variables and values.

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

Arithmetic Operators

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

Assignment Operators

Assignment operators are used to assign values to variables:

Comparison Operators

Comparison operators are used to compare two values:

Logical operators

Logical operators are used to combine conditional statements:

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:

Membership Operators

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

Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Bitwise operators are used to compare (binary) numbers:

```
# **Operator Precedence**

Operator precedence describes the order
performed.
```

Operator Precedence

Operator precedence describes the order in which operations are performed.

Collection Data Types

Lists

A list is an ordered sequence of zero or more object references. Lists support the same slicing and striding syntax as strings

=>The list data type can be called as a function, list() —with no arguments it returns an empty list,

=> An empty list is created using empty brack- ets, [] , and a list of one or more items can be created by using a comma-sepa- rated sequence of items inside brackets.

Assignment

Write a program to increment all the element of list