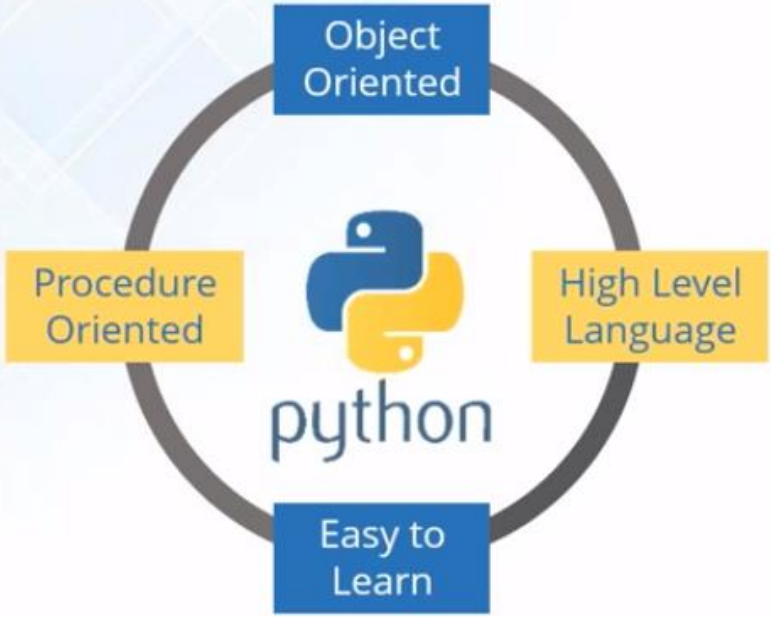
SRM
UNIVERSITY AP
Amaravati

Introduction to Python Programming

Python Introduction

- Python was created by Guido Rossum in 1989 and is very easy to learn.
- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.



Simple & Easy To Learn

Java

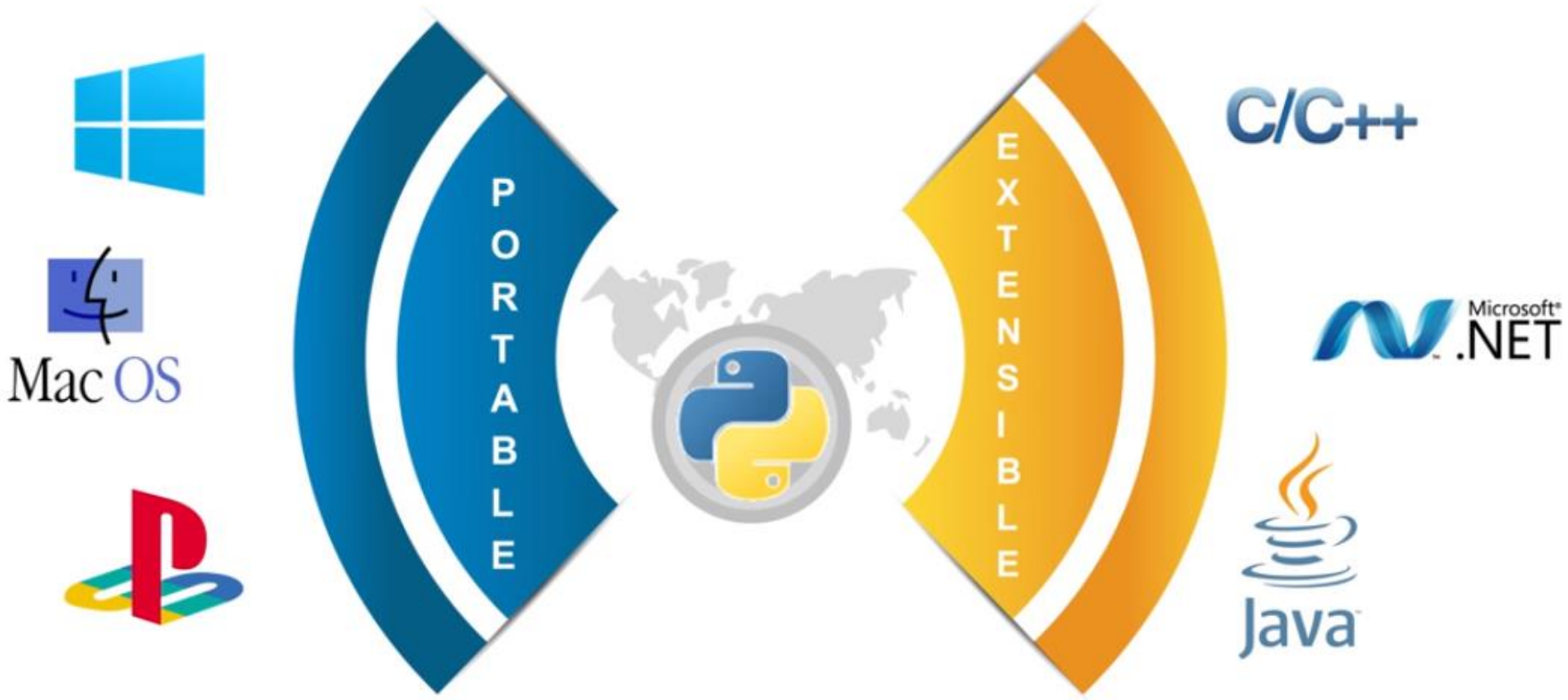
```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

Python

```python
print("Hello, world")
```

It's that **SIMPLE**!

Portable & Extensible



Web Development

- Develop web applications
- Scrape websites

Frameworks

django | Flask | Pylons | WEB2PY

Artificial Intelligence

Libraries

- Scikit-learn
- Keras
- Tensorflow
- Opencv

Computer Graphics

- Graphical User Interface
- Desktop applications
- Game development

Libraries

Tkinter

Jython

wxPython

pygame

Big Data

- Python handles **BIG DATA!**
- Python supports parallel computing
- You can write MapReduce codes in Python

Libraries

PYDOOP | DASK | PySpark

Data Science

- Well-suited for data manipulation & analysis
- Deals with tabular data with heterogeneously-typed columns
- Arbitrary matrix data
- Observational/ statistical datasets

Libraries

NumPy | Pandas | matplotlib | seaborn

Who Uses Python?

Installing Python

- Open a Web browser and go to https://www.python.org/downloads/.

- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.

- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Writing and Executing First Python Program

Step 1: Open an editor.

Step 2: Write the instructions

Step 3: Save it as a file with the filename having the extension .py.

Step 4: Run the interpreter with the command python program_name.py or use IDLE to run the programs.

To execute the program at the *command prompt*, simply change your working directory to C:\Python34 (or move to the directory where you have saved Python) then type python program_name.py.

If you want to execute the program in Python shell, then just press F5 key or click on Run Menu and then select Run Module.

Literal Constants

The value of a literal constant can be used directly in programs. For example, 7, 3.9, 'A', and "Hello" are literal constants.

Numbers refers to a numeric value. You can use four types of numbers in Python program- integers, long integers, floating point and complex numbers.

- **Numbers like 5 or other whole numbers are referred to as *integers*. Bigger whole numbers are called *long integers*. For example, 535633629843L is a long integer.**
- **Numbers like are 3.23 and 91.5E-2 are termed as *floating point numbers*.**
- **Numbers of a + bi form (like -3 + 7i) are *complex numbers*.**

Literal Constants

Strings

A *string* is a group of characters.

- ***Using Single Quotes (')*: For example, a string can be written as 'HELLO'.**
- ***Using Double Quotes (")*: Strings in double quotes are exactly same as those in single quotes. Therefore, 'HELLO' is same as "HELLO".**
- ***Using Triple Quotes (''' ''')*: You can specify multi-line strings using triple quotes. You can use as many single quotes and double quotes as you want in a string within triple quotes.**

Examples:

```
>>> 'Hello'
'Hello'
```

```
>>> "HELLO"
'HELLO'
```

```
>>> '''HELLO'''
'HELLO'
```

Escape Sequences

Some characters (like ", \) cannot be directly included in a string. Such characters must be escaped by placing a backslash before them.

Example:

```
>>> print("The boy replies, \"My name is Aaditya.\"")
The boy replies, "My name is Aaditya."
```

Escape Sequence	Purpose	Example	Output
\\	Prints Backslash	print("\\")	\
\'	Prints single-quote	print("\'")	'
\"	Prints double-quote	print("\"")	"
\a	Rings bell	print("\a")	Bell rings
\f	Prints form feed character	print("Hello\fWorld")	Hello World
\n	Prints newline character	print("Hello\nWorld")	Hello World
\t	Prints a tab	print("Hello\tWorld")	Hello World
\o	Prints octal value	print("\o56")	.
\x	Prints hex value	print("\x87")	+

Variables and Identifiers

Variable means its value can vary. You can store any piece of information in a variable. Variables are nothing but just parts of your computer's memory where information is stored. To be identified easily, each variable is given an appropriate name.

Identifiers are names given to identify something. This something can be a variable, function, class, module or other object. For naming any identifier, there are some basic rules like:

- The first character of an identifier must be an underscore ('_') or a letter (upper or lowercase).
- The rest of the identifier name can be underscores ('_'), letters (upper or lowercase), or digits (0-9).
- Identifier names are case-sensitive. For example, myvar and myVar are not the same.
- Punctuation characters such as @, $, and % are not allowed within identifiers.

Examples of valid identifier names are sum, __my_var, num1, r, var_20, First, etc.

Examples of invalid identifier names are 1num, my-var, %check, Basic Sal, H#R&A, etc.

Assigning or Initializing Values to Variables

In Python, programmers need not explicitly declare variables to reserve memory space. The declaration is done automatically when a value is assigned to the variable using the equal sign (=). The operand on the left side of equal sign is the name of the variable and the operand on its right side is the value to be stored in that variable.

Example:

```
num = 7
amt = 123.45
code = 'A'
pi = 3.1415926536
population_of_India = 10000000000
msg = "Hi"

print("NUM = "+str(num))
print("\n AMT = "+ str(amt))
print("\n CODE = " + str(code))
print("\n POPULATION OF INDIA = " + str(population_of_India))
print("\n MESSAGE = "+str(msg))
```

OUTPUT

```
NUM = 7
AMT = 123.45
CODE = A
POPULATION OF INDIA = 10000000000
MESSAGE = Hi
```

Data Type Boolean

Boolean is another data type in Python. A variable of Boolean type can have one of the two values- True or False. Similar to other variables, the Boolean variables are also created while we assign a value to them or when we use a relational operator on them.

Examples:

>>>Boolean_var = True >>>print(Boolean_var) True	>>> 20 == 30 False	>>>"Python" == "Python" True
>>> 20 != 20 False	>>>"Python"! = "Python3.4" True	>>>30 > 50 False
>>> 90 <= 90 True	>>>87 == 87.0 False	>>>87 > 87.0 False
>>>87 < 87.0 False	>>>87 >= 87.0 True	>>>87 <= 87.0 True

Programming Tip: <, > operators can also be used to compare strings lexicographically.

Input Operation

To take input from the users, Python makes use of the input() function. The input() function prompts the user to provide some information on which the program can work and give the result. However, we must always remember that the input function takes user's input as a string.

Example:

```
name = input("What's your name?")
age = input("Enter your age : ")
print(name + ", you are " + age + " years old")
```

OUTPUT

```
What's your name? Goransh
Enter your age : 10
Goransh, you are 10 years old
```

Comments

Comments are the non-executable statements in a program. They are just added to describe the statements in the program code. Comments make the program easily readable and understandable by the programmer as well as other users who are seeing the code. The interpreter simply ignores the comments.

In Python, a hash sign (#) that is not inside a string literal begins a comment. All characters following the # and up to the end of the line are part of the comment

Example:

```
# This is a comment
print("Hello")  # to display hello
# Program ends here

OUTPUT
Hello
```

Indentation

Whitespace at the beginning of the line is called *indentation*. These *whitespaces or the indentation* are very important in Python. In a Python program, the leading whitespace including spaces and tabs at the beginning of the logical line determines the indentation level of that logical line.

Example:

```
age = 21
    print("You can vote") # Error! Tab at the start of the line
Traceback (most recent call last):
  File "C:\Python34\Try.py", line 2
    print("You can vote")
    ^
IndentationError: unexpected indent
```

Type Conversion

- Data can sometimes be converted from one type to another. For example, the string “3.0” is equivalent to the floating point number 3.0, which is equivalent to the integer number 3
- Functions exist which will take data in one type and return data in another type.
  - `int()` - Converts compatible data into an integer. This function will truncate floating point numbers
  - `float()` - Converts compatible data into a float.
  - `str()` - Converts compatible data into a string.
- Examples:

  `int(3.3)` produces 3
  `str(3.3)` produces “3.3”
  `float(3)` produces 3.0
  `float(“3.5”)` produces 3.5
  `int(“7”)` produces 7
  `int(“7.1”)` throws an ERROR!
  `float(“Test”)` Throws an ERROR!