

Basics Of Python

Master in Python From Zero

What is Python?

- ▶ **Python:** A programming language used to instruct computers to perform tasks. It's based on the CPython interpreter which translates the Python code into something the machine can read.
- ▶ **Interpreted:** Executes code line by line at runtime.
- ▶ **Object-Oriented:** Supports concepts like classes and objects.
- ▶ **High-Level:** Abstracts away complex details, making it easier to code.
- ▶ **Beginner-Friendly:** Simple syntax, ideal for new programmers.
- ▶ **Readability Focus:** Designed for easy reading and understanding.
- ▶ **Efficient:** Reduces the amount of code needed to accomplish tasks.

History Of Python

- ▶ **1980s:** Python was created by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- ▶ **1991:** The first version of Python was released with basic data types and functionality.
- ▶ **1994:** Python 1.0 introduced features like map, lambda, and filter.
- ▶ **2000:** Python 2.0 was released.
- ▶ **2008:** Python 3.0 introduced major updates and improvements.
- ▶ **2022:** Python 3.11, was released with enhanced performance.
- ▶ **2023:** The latest version, 3.12 was released.
- ▶ **Popularity:** Python is now widely used in machine learning, AI, data analysis, web development, and more, offering lucrative career opportunities.

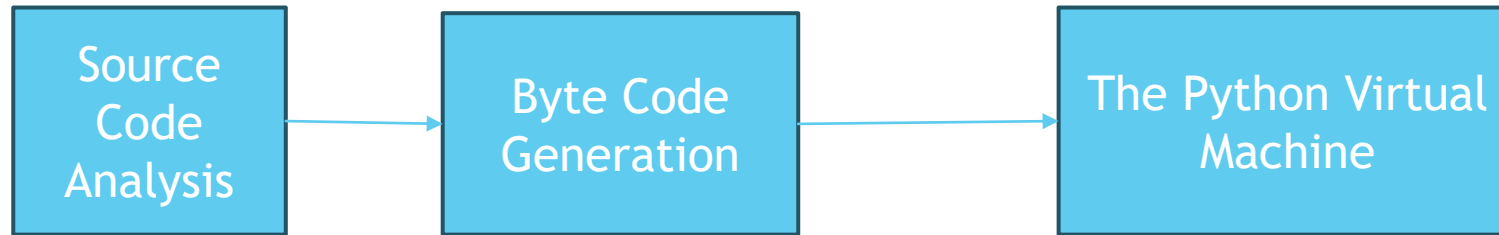
Python Syntax

- ▶ **Syntax:** Standard rules for writing statements in a programming language.
- ▶ **Printing in Python:** Use the `print()` function to display output . Example:
`print("Hello World")`
- ▶ Unlike Python2, which did not require you to put a parenthesis, in Python3, a parenthesis is a must else it will raise a syntax error

Features Of Python

- ▶ *Easy to read and understand*
- ▶ *Interpreted language*
- ▶ *Object-oriented programming language*
- ▶ *Hundreds of libraries and frameworks*
- ▶ *Multi-platform*
- ▶ *Dynamically typed*

How Does the Python Interpreter Work?



- ❑ **CPython:** The default Python interpreter, written in C.
- ❑ **Source Code Analysis:** The interpreter checks syntax and converts source code into tokens (lexical analysis).
- ❑ **Byte Code Generation:** The tokens are compiled into bytecode, which can be saved as .pyc files.
- ❑ **PVM Execution:** The Python Virtual Machine (PVM) converts bytecode into executable machine code, which is then executed.

What Is The Difference Between .Py And .Pyc Files?

| Feature | .py Files | .pyc Files |
|----------------|---|--|
| Content | Contains Python source code | Contains compiled Python bytecode. |
| Human-readable | Yes | No |
| Editability | Can be edited with any text editor or IDE | Not meant to be edited |
| Generation | Created manually by the programmer | Automatically generated by the Python interpreter when a .py file is run |
| Execution | Must be compiled to bytecode by the Python interpreter before execution | Directly executable by the Python virtual machine |
| Purpose | For writing and editing Python code | For faster execution and to save bytecode |

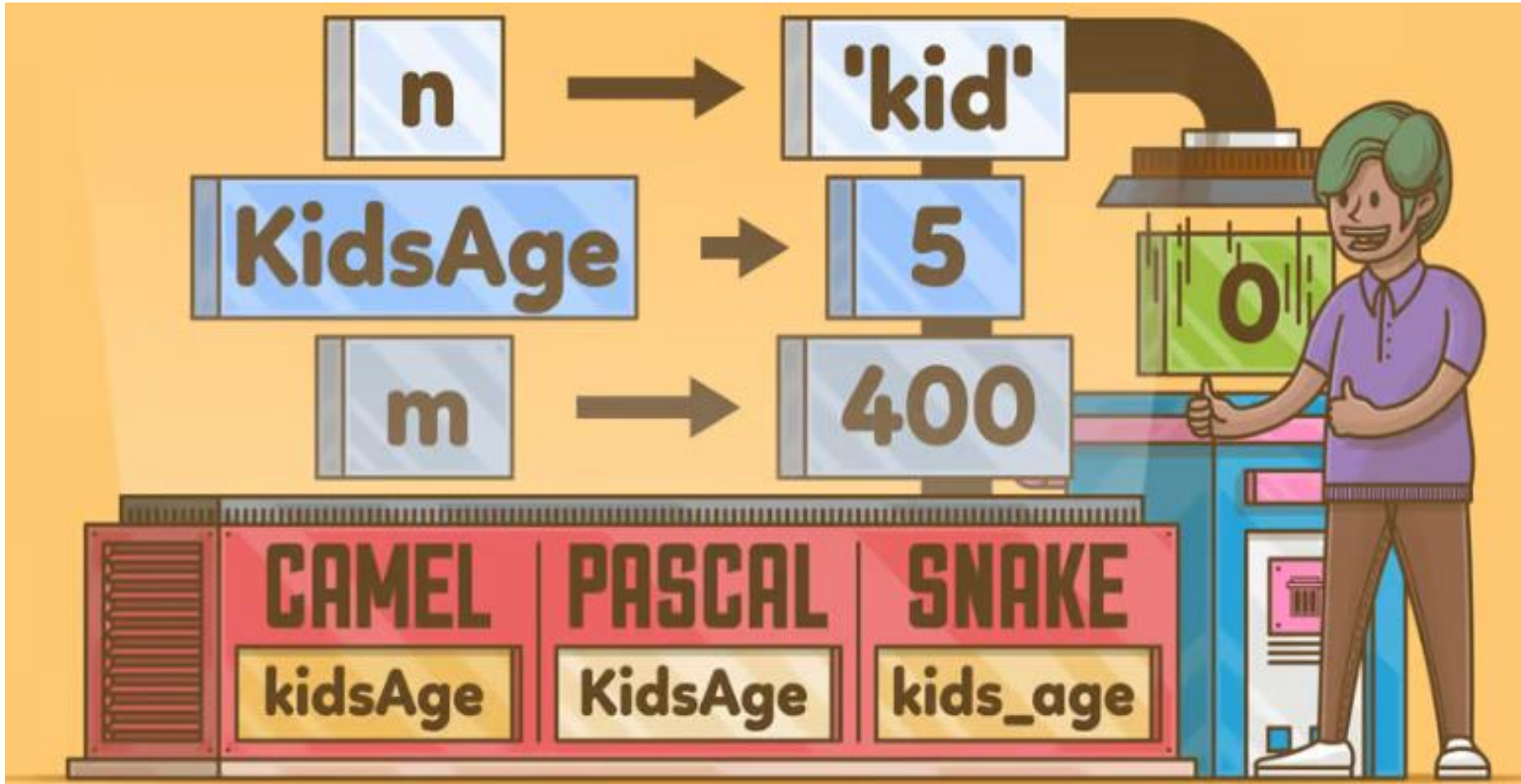
Python First Program

- ▶ `Print("Hello,World!")`
- ▶ Python `print()` function prints the message to the screen or any other standard output device.
- ▶ **How `print()` works in Python?**
- ▶ You can pass variables, strings, numbers, or other data types as one or more parameters when using the `print()` function. Then, these parameters are represented as strings by their respective `str()` functions. To create a single output string, the transformed strings are concatenated with spaces between them.

Python Variables

- ▶ Python Variable are containers that store values.
- ▶ In Python, variables need not be declared or defined in advance, as is the case in many other programming languages.
- ▶ To create a variable, just assign it a value and start using it. An assignment is done with a single equals sign (=). Ex. N=300

Rules to define Variables



Python Keywords

- ▶ There is one more restriction on identifier names. The Python language reserves a small set of [keywords](#) that designate special language functionality. No object can have the same name as a reserved word.
- ▶ In Python 3, there are 33 reserved keywords you can see in the next slide. Each keyword has a different meaning and we can not use the keyword as a variable name.

Reserve Words

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

Python Literals

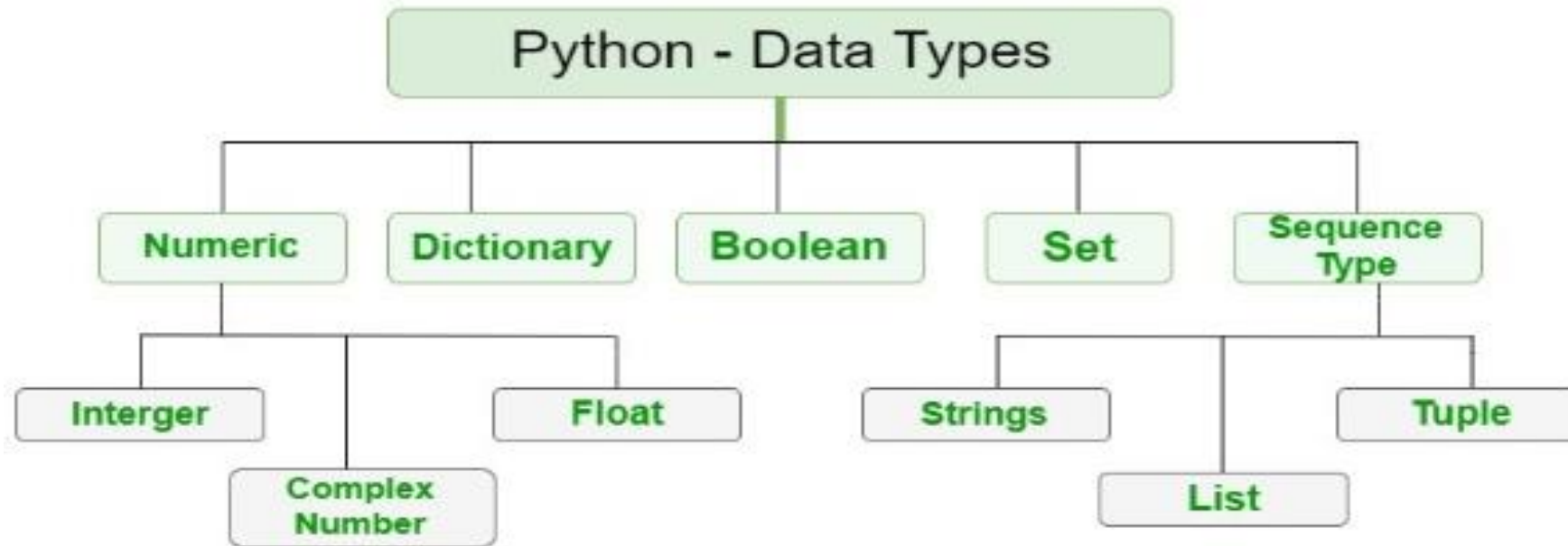
- ▶ Literals are representations of fixed values in a program. They can be numbers, characters, or strings, etc.
- ▶ In above example we can see site_name is variable and 'apple.com' is a literal.

```
site_name = 'apple.com'
```

Types of Literals :

1. Integer Literals
2. Floating-Point Literals
3. Complex Literals
4. Python String Literals

Data Types in Python



Comments in Python

- ▶ Comments are hints that we add to our code to make it easier to understand. Python comments start with #.
- ▶ Comments are completely ignored and not executed by code editors.
- ▶ **Single-line Comment** : We use the **hash (#)** symbol to write a single-line comment.
- ▶ **Multiline Comments** : we can achieve the same effect by using the hash (#) symbol at the beginning of each line.
- ▶ We can also use multiline strings as comments like `''' '''`.
- ▶ If we encounter an error while running a program, instead of removing code segments, we can comment them out to prevent execution.

Why Use Comments?

- For future references, as comments make our code readable.
- For debugging.
- For code collaboration, comments help peer developers to understand each other's code.

Note: Comments are not and should not be used as a substitute to explain poorly written code. Always try to write clean, understandable code, and then use comments as an addition.

Operators in Python

- ▶ Operators are special symbols that perform operations on [variables](#) and values.

```
print(5 + 6)  # 11
```

Run Code

Here, `+` is an operator that adds two numbers: **5** and **6**.

Types of Python Operators

1. Arithmetic Operators - Arithmetic operators are used to perform mathematical operations like `+, -, *, /, //, %, **`
2. Assignment Operators - Assignment operators are used to assign values to variables like `=, +=, -=, *=, /=, %=, //=, **=`
3. Comparison Operators - Comparison operators compare two values/variables and return a boolean result: `True` or `False` like `==, !=, >=, <=, >, <`
4. Logical Operators - Logical operators are used to check whether an expression is `True` or `False` like `and, Or, Not`
5. Bitwise Operators - Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit like `&, |, ~`.
6. Special Operators - Python language offers some special types of operators like the **identity** operator and the **membership** operator.

Bitwise Operators

Bitwise AND

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

AND (&)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Bitwise OR

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

OR (|)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Bitwise NOT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

NOT (~)

Precedence of Python Operators

- ▶ The combination of values, [variables](#), [operators](#), and [function](#) calls is termed as an expression. The Python interpreter can evaluate a valid expression.

```
>>> 5 - 7  
-2
```

Here `5 - 7` is an expression. There can be more than one operator in an expression.

- ▶ To evaluate these types of expressions there is a rule of precedence in Python. It guides the order in which these operations are carried out.

For example, multiplication has higher precedence than subtraction.

```
# Multiplication has higher precedence  
# than subtraction  
>>> 10 - 4 * 2  
2
```

But we can change this order using parentheses `()` as it has higher precedence than multiplication.

```
# Parentheses () has higher precedence  
>>> (10 - 4) * 2  
12
```