



# COMPUTING FUNDAMENTALS USING PYTHON

---

**Archana A**

Department of Computer Applications

# COMPUTING FUNDAMENTALS USING PYTHON

---

## Python – Basics

**Archana A**

Department of Computer Applications

- A program in python has number of statements.
- These statements are followed by the computer one after another in a sequence Called as “executing a program” or “running a program”.
- Python distinguishes uppercase and lowercase characters – like ‘A’ and ‘a’.
- Most of the words used in Python are in **lower case**.

- Language has grammar rules called **syntax**.
- If the syntax is violated, the Python translator gives an **error**.
  - A sequence of symbols(characters) is called a **string**.
  - A string which does not change is referred to as a **string constant** or **string literal**.

- String has to be surrounded by **quotes** – **single or double** should be same in the beginning as well as the end.
  - If quotes are missed at one end or not properly matched, then the translator gives an error.
- All statements unless special construct is used should start from the first column.
  - If the rule is not followed, the translator gives an error

- Instructions that a Python interpreter can execute are called **statements**.
- The end of a statement is marked by a newline character.
- A statement can be extended over multiple lines with the line continuation character (\).
- Line continuation is implied inside parentheses ( ), brackets [ ] and braces { }

```
# multi-line statement
result = 2 + 3 * \
5 - 5 + 6 - \
3 + 4
print(result)
```

```
# Initializing a
# list / mathematical expression
# using the Implicit multi-line
# statement.
list = [5,
        4, 3, 2, 1
       ]

add = (50 +
      40 -
      52)
```

- No braces to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentation, which is rigidly enforced.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

- Statements contained within the [], {} or () brackets do not need to use the line continuation character that is \

```
# Initializing a  
# list / mathematical expression  
# using the Implicit multi-line  
# statement.  
list = [5,  
        4, 3, 2, 1  
]  
  
add = (50 +  
       40 -  
       52)
```

- A **hash sign (#)** that is not inside a string literal begins a comment.
- All characters after the # and up to the physical line end are part of the comment and the Python interpreter ignores them.

# This is a comment

- A line containing whitespace, possibly with a comment, is known as a **blank line** and **Python totally ignores it**.
- In an interactive interpreter session, an empty physical line should be entered to terminate a multiline statement

- The **semicolon** ( **;** ) allows multiple statements on the single line given that neither statement starts a new code block
- For example:
  - `x =4; y = 6; s = x+y; print(s) #(correct)`
  - `x=3; y=5; if x<y: #(wrong)`

- A group of individual statements, which make a single code block are called **suites** in Python.
- Compound or complex statements, such as if, while, def, and class require a header line and a suite.

# COMPUTING FUNDAMENTALS USING PYTHON

---

## Python – Keywords, Identifiers, Variables, Literals

**Archana A**

Department of Computer Applications

- Keywords are the **reserved words** in Python.
- Keywords **cannot be** used as 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 3.3
- All the keywords except **True**, **False** and **None** are in lowercase and they must be written as it is.

```
>>>import keyword
```

```
>>>keyword.kwlist
```

### 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    |        |

- All the keywords except **True**, **False** and **None** are in lowercase and they must be written as it is.

- Identifier is the name given to entities like class, functions, variables etc. in Python.
- It helps differentiating one entity from another.

- 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 (\_)**.
  - Examples - myClass, var\_1 and print\_this\_to\_screen, all are valid example.
- An identifier cannot start with a digit.

Example

- 1variable is invalid
- variable1 is perfectly fine.

- Keywords **cannot** be used as **identifiers**.
- Special symbols like !, @, #, \$, % etc. **cannot** be used.
- Identifiers can be of any length.

Naming (a-z, A-Z, 0-9, \_)

Starts with alphabet or \_

Case sensitive (total, Total, TOTAL)

Length

- Keywords cannot be used as identifiers

Differences: x, \_x (protected), \_\_x (private), \_\_x\_\_ (magic / predefined)

- A variable is a way of referring to a memory location used by a computer program.
- A variable is a symbolic name for this physical location.
- This memory location contains values, like numbers, text or more complicated types.
- Python is a type inferred language, it can automatically infer the type of the variable.

- A variable is created the moment it is first assigned a value.
- Variables **do not** need to be **declared with any particular type** and can even change type after they have been set.
- A single value can be assigned to several variables simultaneously.

- One of the most fundamental concepts in programming is that of a **variable**.
- A variable is “a name that is assigned to a value,” as shown below,

`n = 5`                      variable n is assigned the value 5

Thus, whenever variable n appears in a calculation, it is the current value of n that is used, as in the following,

`n + 20`    (5 + 20)

If variable n is assigned a new value, then the same expression will produce a different result,

`n = 10`  
`n + 20`    (10 + 20)

- A constant is a type of variable whose value cannot be changed.
- They are containers that hold information which cannot be changed later.

- To tell other programmers that a given value should be treated as a constant, we must use a widely accepted naming convention for the constant's identifier or name.
- We should write the name in capital letters with underscores separating words

```
PI = 3.14
MAX_SPEED = 300
DEFAULT_COLOR = "\033[1;34m"
WIDTH = 20
API_TOKEN = "593086396372"
BASE_URL = "https://api.example.com"
DEFAULT_TIMEOUT = 5
```

### Assigning value to constant in Python:

- In Python, constants are usually declared and assigned in a module. Here, the module is a new file containing variables, functions, etc which is imported to the main file. Inside the module, constants are written in all **capital letters** and underscores separating the words.
- Example: Create a **constant.py**:
- $\text{PI}=3.14$
- $\text{GRAVITY}=9.8$

### Assigning value to constant in Python:

- Create another file main.py

Import constant

Print(constant.PI)

Print(constant.GRAVITY)

Output:

3.14

9.8

- Literal is a raw data given in a variable or constant.
- Python allows:
  - Numeric Literals
  - String literals
  - Boolean literals
  - Special literals - None

- Numeric Literals are immutable (unchangeable).
- Numeric literals can belong to 3 different numerical types
  - Integer
  - Float
  - Complex

- How to use Numeric Literals

```
a = 0b1010 #Binary Literals
b = 100 #Decimal Literal
c = 0o310 #Octal Literal
d = 0x12c #Hexadecimal Literal
```

```
#Float Literal
```

```
float_1 = 10.5
```

```
float_2 = 1.5e2
```

```
#Complex Literal
```

```
x = 3.14j
```

```
print(a, b, c, d)
```

```
print(float_1, float_2)
```

```
print(x, x.imag, x.real)
```

# COMPUTING FUNDAMENTALS USING PYTHON

## Literals

---

- We assigned integer literals into different variables. Here, a is binary literal, b is a decimal literal, c is an octal literal and d is a hexadecimal literal.
- When we print the variables, all the literals are converted into decimal values.
- 10.5 and 1.5e2 are floating-point literals. 1.5e2 is expressed with exponential and is equivalent to  $1.5 * 10^2$ .
- We assigned a complex literal i.e 3.14j in variable x. Then we use **imaginary** literal (x.imag) and **real** literal (x.real) to create imaginary and real parts of complex numbers.

- A string literal is a sequence of characters surrounded by quotes.
- **Single, double or triple quotes** are used for a string.
- A character literal is a single character surrounded by single or double quotes.

```
strings = "This is Python"
char = "C"
multiline_str = """This is a multiline string with more than one line code."""
unicode = u"\u00dcnic\u00f6de"
raw_str = r"raw \n string"

print(strings)
print(char)
print(multiline_str)
print(unicode)
print(raw_str)
```

- Output: .

```
This is Python
C
This is a multiline string with more than one line code.
Ünicöde
raw \n string
```

- Boolean literal can have any of the two values: **True** or **False**.

```
x = (1 == True)
y = (1 == False)
a = True + 4
b = False + 10

print("x is", x)
print("y is", y)
print("a:", a)
print("b:", b)
```

- Output: .

```
x is True
y is False
a: 5
b: 10
```

Python contains one special literal i.e. **None**. We use it to specify that the **field has not been created**.

**Example : How to use special literals in Python?**

```
drink = "Available"
food = None

def menu(x):
    if x == drink:
        print(drink)
    else:
        print(food)

menu(drink)
menu(food)
```

**Output**

```
Available
None
```

- In the example program, we define a **menu** function. Inside **menu**, when we set the argument as **drink** then, it displays **Available**. And, when the argument is **food**, it displays **None**.



**THANK YOU**

---

**Ms. Archana A**  
Assistant Professor  
Department of Computer Science  
**archana@pes.edu**  
+91 80 6666 3333 Extn 392