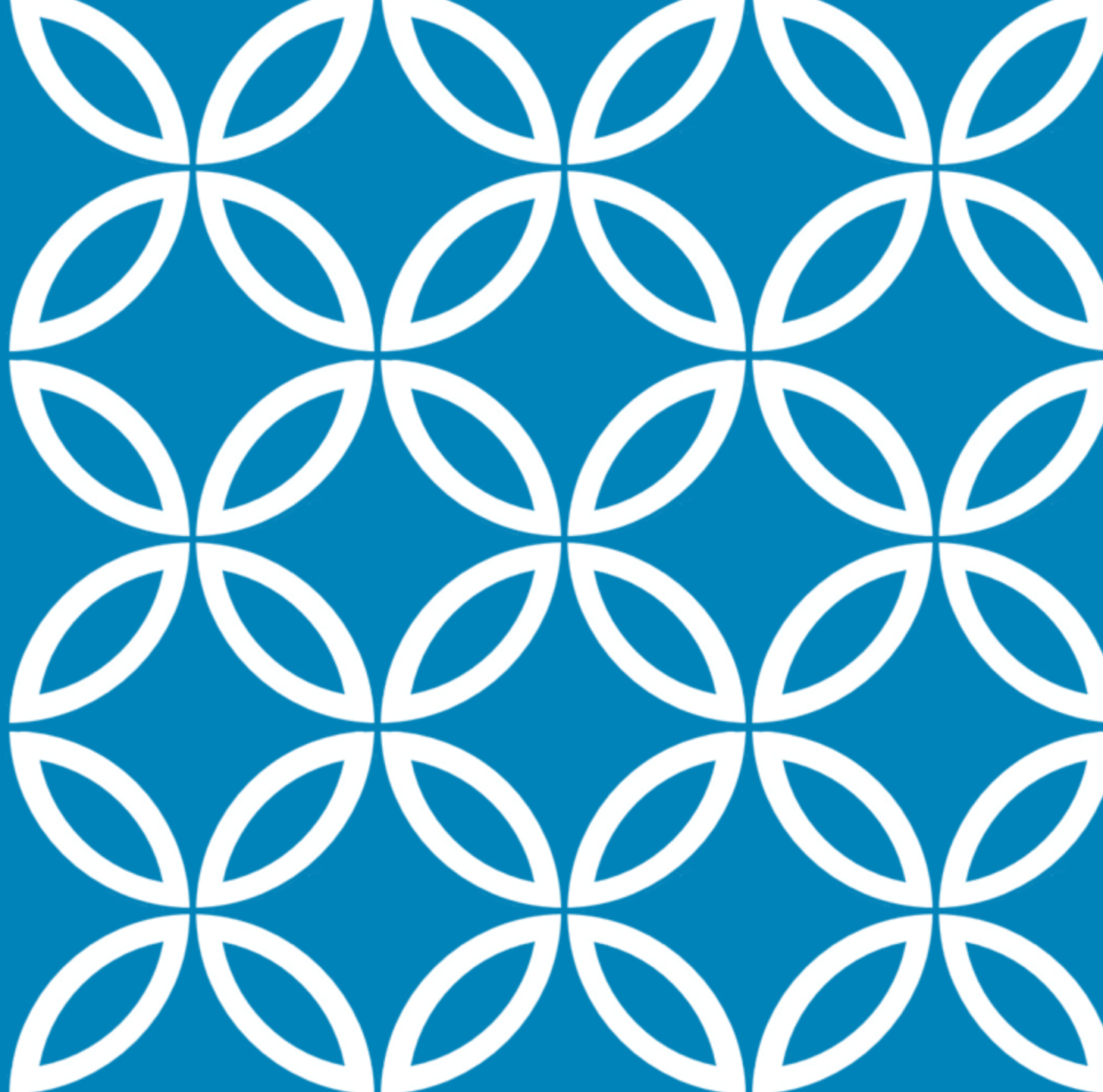


# FUNDAMENTAL CONCEPTS IN PYTHON

---



# CHARACTER SET

- Character set defines the valid characters that can be used in a source program .Python uses the character set as the building block to form the basic program elements such as variables, identifiers ,constants, expressions etc.
- Python uses the *traditional ASCII character set*. The latest version also recognizes the *Unicode character set*.
- The *ASCII character set* is a subset of the *Unicode character set*. The main limitation of ASCII character is its inability to represent more than  $256(=2^8)$ .
- Unicode character set, a set that use 2 bytes (16 bits) per character..

# IDENTIFIERS

❑ **Identifier** is the name given to entities like class, functions, variables etc. in Python. It helps differentiating 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 my Class, var\_1 and print\_this\_to\_screen, all are valid example.**
- An identifier must start with a letter or an underscore.
- It cannot start with a digit.
- **variable and \_variable are valid identifiers but 1variable is invalid.**
- An identifier cannot be a keyword .Keywords, also called reserved words, have special meanings in Python. For example, import is a keyword, which tells the Python interpreter to import a module to the program.
- An identifier can be of any length.

# KEYWORDS

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

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

# VARIABLES

Variables are used to reference values that may be changed in the program.

Variable is a name of the memory location where data is stored. Once a variable is stored that means a space is allocated in memory.

- We need not to declare explicitly variable in Python as we do in c, c++, and java.

```
int a, b;  
float c, d;
```

When we assign any value to the variable that variable is declared automatically. The statement for assigning a value to a variable is called an assignment statement. In Python, the equal sign (=) is used as the assignment operator. The syntax for assignment statements is as follows:

Variable = expression

# DATA TYPE

A data type is a set of values, and a set of operators that may be applied to those values.

For example, the integer data type consists of the set of integers, and operators for addition, subtraction, multiplication, and division, among others. Integers, floats, complex numbers and strings are part of a set of predefined data types in Python called the built-in types.

There are various data types in Python. Some of the important types are listed below.

## **Numbers**

Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.

- a) **Integer (signed)** -Numbers (can be both positive and negative) with no fractional part.e.g. 100
- b) **Float pointing** - Real numbers with both integer and fractional part e.g. 100.90
- c) **Complex** - In the form of  $a+bj$  where a forms the real part and b forms the imaginary part of complex number. e.g.  $3+4j$

## **Strings**

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings.

```
s = "This is a string"
```

# EXPRESSION

An expression represents a computation involving values, variables, and operators that, taken together, evaluate to a value.

You can use a variable in an expression. A variable can also be used in both sides of the `=` operator.

For example, consider the following code:

```
y = 1 # Assign 1 to variable y
```

```
Radius = 1.0 # Assign 1.0 to variable Radius
```

```
x = 5 * (3 / 2) + 3 * 2 # Assign the value of the  
expression to x
```

```
x = y + 1 # Assign the addition of y and  
1 to x
```

```
area = radius * radius * 3.14159 # Compute  
area
```

# SIMULTANEOUS ASSIGNMENT

Python also supports simultaneous assignment in syntax like this:

```
var1, var2, var3 = exp1, exp2, exp3
```

It tells Python to evaluate all the expressions on the right and assign them to the corresponding variable on the left simultaneously. Swapping variable values is a common operation in programming and simultaneous assignment is very useful to perform this operation.

Consider two variables: `x` and `y`. How do you write the code to swap their values? A common approach is to introduce a temporary variable as follows:

```
x = 1
y = 2
temp = x           # Save x in a temp variable
x = y              # Assign the value in y to x
y = temp           # Assign the value in temp to y
```

But you can simplify the task using the following statement to swap the values of `x` and `y`.

```
x, y = y, x           # Swap x with y
```



# MIXED TYPE EXPRESSION

A mixed-type expression is an expression containing operands of different type. The CPU can only perform operations on values with the same internal representation scheme, and thus only on operands of the same type.

Operands of mixed-type expressions therefore must be converted to a common type. Values can be converted in one of two ways—by implicit (automatic) conversion, called coercion, or by explicit type conversion

## *Coercion vs. Type Conversion*

**Coercion** is the implicit (automatic) conversion of operands to a common type. Coercion is automatically performed on mixed-type expressions only if the operands can be safely converted, that is, if no loss of information will result. The conversion of integer 2 to floating-point 2.0 below is a safe conversion—the conversion of 4.5 to integer 4 is not, since the decimal digit would be lost,

a=2  
b=4.5  
c=a + b

$$2 + 4.5 \rightarrow 2.0 + 4.5 \rightarrow 6.5$$

(automatic conversion of int to float)

Type conversion is the **explicit conversion** of operands to a specific type. Type conversion can be applied even if loss of information results.

$$2 + \text{int}(4.5) \rightarrow 2 + 4 \rightarrow 6$$

# COMMENTS

Python, comments are preceded by a pound sign (#) on a line, called a line comment, or enclosed between three consecutive single quotation marks (") on one or several lines, called a paragraph comment.

When the Python interpreter sees #, it ignores all text after # on the same line. When it sees ", it scans for the next " and ignores any text between the triple quotation marks.

```
# This program displays Welcome to Python
```

```
" This program displays Welcome to Python and  
    Python is fun  
"
```

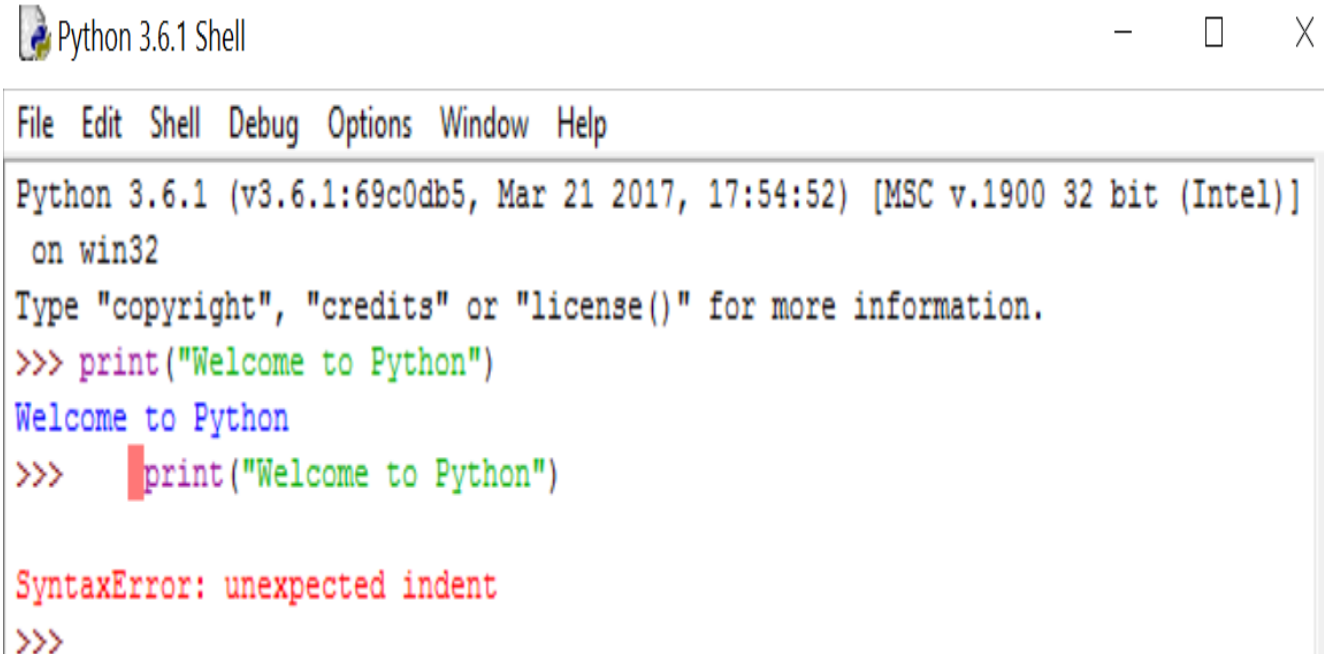
# INDENTATION

Indentation matters in Python. Note that the statements are entered from the first column in the new line. The Python interpreter will report an error if the program is typed as follows:

```
# Display two messages
```

```
    print("Welcome to Python")
```

```
    print("Python is fun")
```



The screenshot shows a Python 3.6.1 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The command prompt displays the Python version and build information. The user enters a comment followed by two indented print statements. The first statement is correctly indented and executes. The second statement is also indented but causes a SyntaxError because it is not part of a block. The error message 'SyntaxError: unexpected indent' is displayed in red.

```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Welcome to Python")
Welcome to Python
>>>     print("Welcome to Python")

SyntaxError: unexpected indent
>>>
```

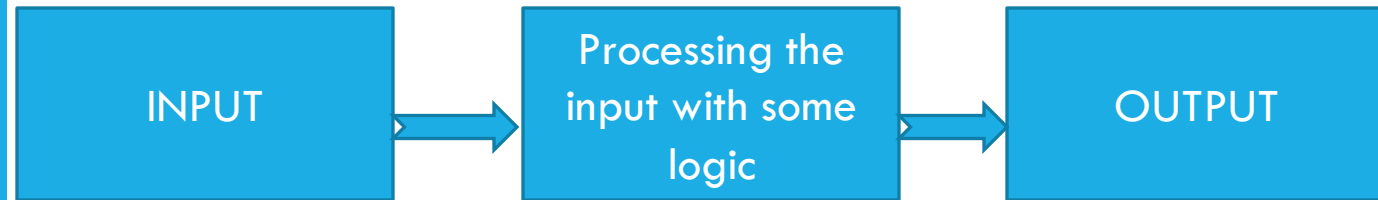
# INPUT AND OUTPUT

The purpose of a computer is to process data and return results. It means that first of all, we should provide data to the computer.

The data given to the computer is called input. The results returned by the computer are called output.

So, we can say that a computer takes input, processes that input and produces the output.

To provide input to a computer, Python provides some statements which are called Input statements. Similarly, to display the output, there are Output statements available in Python. We should use some logic to convert the input into output.



# OUTPUT STATEMENT

To display output or results, Python provides the `print()` function.

```
print('Hello')  
Hello
```

```
print("This is the \n first line")  
This is the  
first line
```

```
a, b = 2, 4  
print(a)  
2  
print(a, b)  
2 4
```

# INPUT STATEMENT

To accept input from keyboard, Python provides the **input()** function.

This function takes a value from the keyboard and returns **it as a string**.

```
str = input()      #this will wait till we enter a string
print(str )
```

```
Raj kumar        #enter this string
```

```
Raj kumar        # output
```

```
str = input('Enter your name:')
print(str)
```

```
Enter your name: Raj kumar    # input
Raj kumar                     # output
```

# PRACTICE QUESTION 1

What will be the output of the code?

```
x= input("enter the value")
```

```
# Let x= 12
```

```
y=x+2
```

```
print(y)
```

(a) 14

(b) "12" + 2

(c) "12" + "2"

(d) Error

# INPUT STATEMENT

```
str = input('Enter a number:')  
x = int(str)    #str is converted into int  
print(x)  
Enter a number: 125  
125
```

```
x = int(input('Enter a number:'))  
print(x)  
Enter a number: 125  
125
```



## PRACTICE QUESTION 2

What are keywords in Python?

- a) Keywords are reserved words that are used to construct instructions.
- b) Keywords are used to calculate mathematical operations.
- c) Keywords are used to print messages like "Hello World!" to the screen.
- d) Keywords are the words that we need to memorize to program in Python.