

The title 'PYTHON FUNDAMENTALS' is centered within a white, multi-lobed circular shape. The background is a solid orange color, and a dark brown vertical bar is on the left side. The text is in a bold, dark brown, sans-serif font.

PYTHON FUNDAMENTALS

PYTHON 3.6.5

PYTHON CHARACTER SET

Character set is a set of valid characters that a language can recognize.

A character represents any letter, digit or any other symbol.

Category	Example
Letters	A-Z, a-z
Digits	0-9
Special Symbols	Space + - * / ** \ () [] {} // = != == < > . ‘ “ ” “”” , ; : % ! & # <= >= @ >>> << >> _(underscore)
White Spaces	Blank space, tabs, carriage return, new line, form-feed
Other Characters	All other ASCII and Unicode characters

TOKEN

The smallest individual unit in a program is known as token.

Python has following tokens:-

- Keyword
- Identifiers (Names)
- Literals
- Operators
- Punctuators

KEYWORDS

- A keyword is a word having special meaning reserved by the programming language.

Keywords in Python 3.3 programming language

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	

MORE ON KEYWORDS....

- We cannot use a keyword as variable name, function name or any other identifier.
- In Python, keywords are case sensitive.
- All the keywords except True, False and None are in lowercase and they must be written as it is. The list of all the keywords are given below.

PYTHON 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 myClass, var_l and print_this_to_screen, all are valid example.
- An identifier cannot start with a digit. lvariable is invalid, but variablel is perfectly fine.
- Keywords cannot be used as identifiers.
- We cannot use special symbols like !, @, #, \$, % etc. in our identifier.
- Identifier can be of any length.

LITERALS

Literals are data items that have a fixed values.

String Literals

- Single Line String
- Multi Line String
- Raw String
- Unicode String

- Numeric Literals

- Integer
- Float
- Complex

- Boolean Literals
- Special Literals
- Literal Collection
 - List
 - Tuple
 - Dictionary
 - Set

STRINGS

The text enclosed in quotes forms a string literals. A string literal is a sequence of characters surrounded by quotes. We can use both single, double or triple quotes for a string. And, a character literal is a single character surrounded by single or double quotes.

Single Line String	"hello world"
Multi Line String	"""Allahabad Uttar Pradesh"""
Raw String	r"raw \n string"
Character	"C"
Unicode string	u"\u00dcnic\u00f6de", u"\u0930\u093E\u0940\u0935"

EXAMPLE

- `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)`

NUMERIC LITERALS

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

- Integer
 - Decimal
 - Binary Literal
 - Octal
 - Hexadecimal
- Float
- Complex

EXAMPLE

- `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)`

BOOLEAN LITERALS

- There are two kinds of Boolean literal: **True** and **False**.
- Boolean literal contains 1 as true value and 0 as false value.
- Example:-

```
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)
```

SPECIAL LITERAL

- Python contains one special literal i.e. **None**
- None literal represents absence of a value.
- We use it to specify to that field that is not created.

Example

```
drink = "Available"
food = None
def menu(x):
    if x == drink:
        print(drink)
    else:
        print(food)
menu(drink)
menu(food)
```

OPERATORS

- Operators are tokens that trigger some computation when applied to variables and other objects in an expression.
- Operator requires operands to work upon.
- Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

For example:

```
>>> 2+3
```

It requires two types of operators:-

Unary Operator (+, -, ~, not)

Binary Operator

Arithmetic Operator (+, -, *, /, %, **, //)

Relational Operator (<, >, <=, >=, ==, !=)

Logical Operator (and, or, not)

Bitwise Operator (&, ^, |, <<, >>)

Assignment Operator (=, +=, -=, *=, /=, %=, **=, //=)

Identity Operator (is, not is)

Membership Operator (in, not in)

PUNCTUATORS

- Punctuators are symbols that are used in programming language to organize sentence structures and indicate the rhythm and emphasis of expressions, statements and program structure.
- Most common punctuators in python are-
`' " # \ () [] { } @ , : . = ;`

PYTHON PROGRAM STRUCTURE

- Python program contains various components like:-

1. Expressions
2. Statements
3. Comments
4. Function
5. Blocks and Indentation

1. Expressions

An expression is any legal combination of symbols that represents a value.

e.g., `a = 15`

2. Statement

- Instructions that a Python interpreter can execute are called statements.
- A statement is a programming instruction that does something i.e., some action takes place.

`a=10`
`Print(a);` } these are statements

3. Comments

- Comments are the additional readable information to clarify the source code. Comments are ignored by the python interpreter but it can be read by the programmer.
- Comments in Python begin with symbol `#` or `"""`. Comments enclosed in triple quotes are called docstrings.

Single Line Comment:

- In Python, we use the hash (`#`) symbol to start writing a comment.

```
#This is a comment
```

```
#print out Hello
```

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

Multi-line comments

- Another way of doing this is to use triple quotes, either `'''` or `"""`.

```
"""This is also a
```

```
perfect example of
```

```
multi-line comments"""
```

```
print("Hello")
```

4. Functions

A function is a code that has a name and it can be reused by specifying its name in the program, where needed.

Example

```
drink = "WATER"
```

```
food = "EAT"
```

```
Sit = "TABLE"
```

```
def menu(x):
```

```
    if x == drink:
```

```
        print(drink);
```

```
        print("Prints only if condition is TRUE");
```

```
    else:
```

```
        print(food);
```

```
        print("Prints only if condition is FALSE");
```

```
Print(sit);
```

```
menu(drink)
```

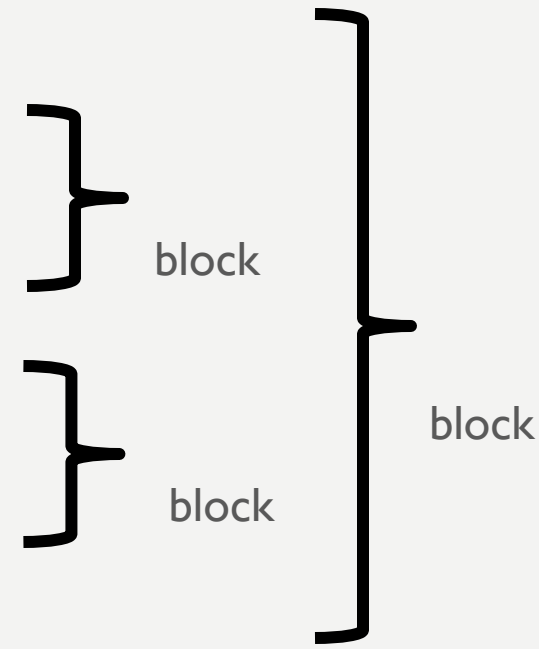
5. Blocks and Indentation

BLOCKS

A group of statements which are part of another statement or a function are called block or code block or suite in Python.

Example

```
def menu(x):  
    if x == drink:  
        print(drink);  
        print("Prints only if condition is TRUE");  
    else:  
        print(food);  
        print("Prints only if condition is FALSE");  
Print(sit);  
menu(drink)
```



block

block

block

INDENTATION

- Python uses indentation to create blocks of code..
- A code block (body of a function, loop, class etc.) starts with indentation and ends with the first unindented line.
- The amount of indentation is up to you, but it must be consistent throughout that block.
- Generally four whitespaces are used for indentation and is preferred over tabs.

```
c=a+b
```

```
if c<50:
```

```
    print("less than 50");
```

```
    b=b*2
```

```
    a=a+10
```

```
else:
```

```
    print("greater than 50");
```

```
    b=b*2
```

```
    a=a+10
```

VARIABLES IN PYTHON

- A variable in python represents named location that refers to a value whose values can be used and processed during program run.
- A named labels whose values can be used and processed during program run are called Variables.
 - `a=10` single variable
 - `a, b, c = 5, 3.2, "Hello"` multiple variable having multiple values
 - `x = y = z = "same"` multiple variable having single value
- In python a variable is created when we first assign a value to it.

- **Python Variable Scope**
- Scope is the portion of the program from where a namespace can be accessed directly without any prefix. Scope of the current function which has local names
- Scope of the module which has global names
- If there is a function inside another function, a new scope is nested inside the local scope.
- **Example of Scope and Namespace in Python**

Example 1:

```
def outer_function():  
    a = 20  
    def inner_function():  
        a = 30  
        print('a =',a)  
    inner_function()  
    print('a =',a)  
a = 10  
outer_function()  
print('a =',a)
```

Example 2:

```
def outer_function():  
    global a  
    a = 20  
    def inner_function():  
        global a  
        a = 30  
        print('a =',a)  
    inner_function()  
    print('a =',a)  
a = 10  
outer_function()  
print('a =',a)
```

PYTHON OUTPUT

- Python provides print() function for output

Printing a string

```
print("Hello World");
```

Printing variable

```
a=10
```

```
print(a);
```

Printing multiple variables

```
a=10
```

```
b=20
```

```
c=30
```

```
print("Values of a, b and c =", a, b, c);
```

PYTHON INPUT

- In Python input() functions read data from keyboard as string, irrespective of whether it is enclosed with quotes (" or "") or not.

Taking input from keyboard at runtime

```
age=input("Enter you age:")  
print("Your age =",age);
```

Input() function always returns a value of string type.

To convert the string value into integer type python offers two functions int() and float(). Both functions are used to convert string type input to integer type.

Example I:

```
Name = input("What is your name : ")  
print(Name);
```


Example 2:

```
A=input("Enter First number : ")  
B=input("Enter Second number : ")  
A=int(A)  
B=int(B)  
C=A+B  
print(C);
```

We can also write the above program as

```
A=int(input("Enter First number : "))  
B=int(input("Enter Second number : "))  
C=A+B  
print(C);
```