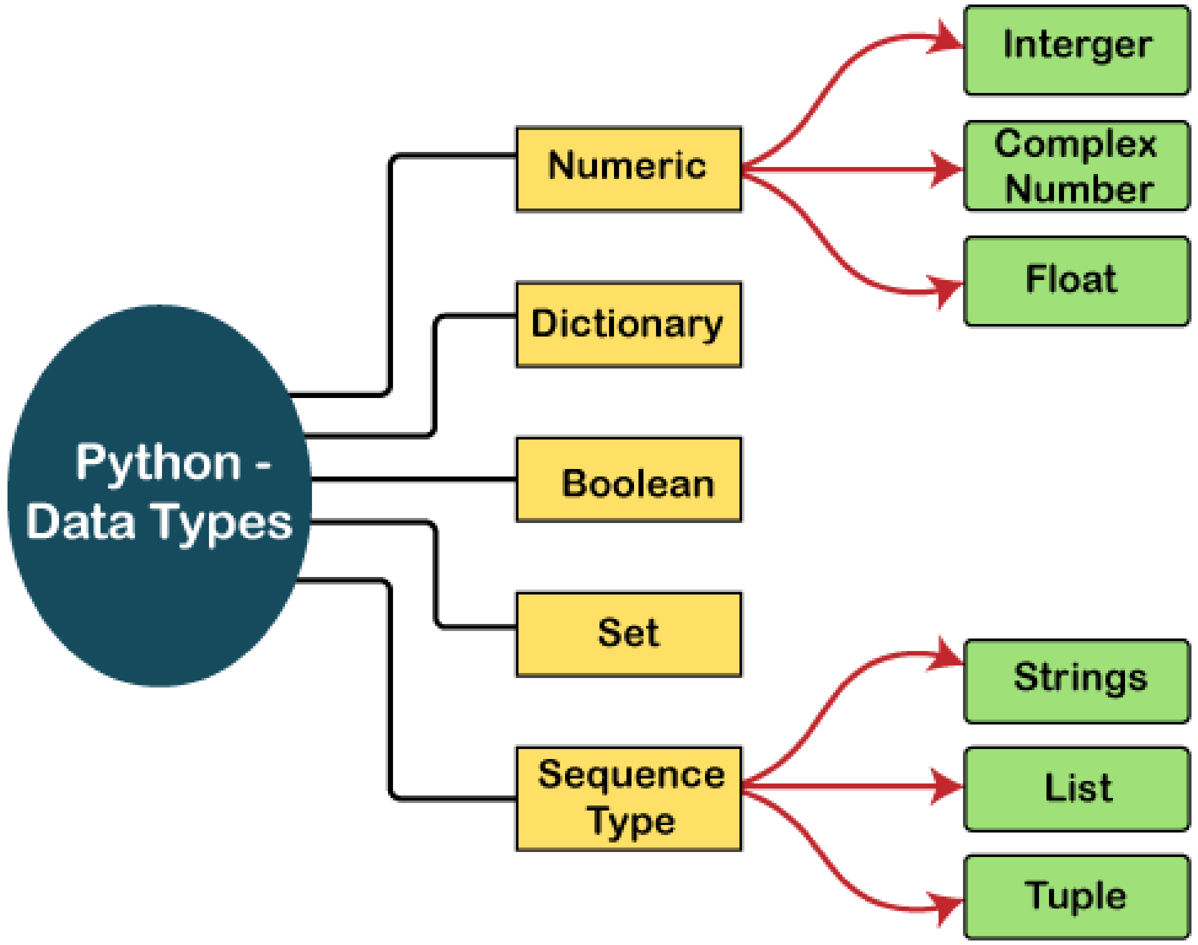
Python Fundamentals

Learning Objectives

- To understand Python fundamentals

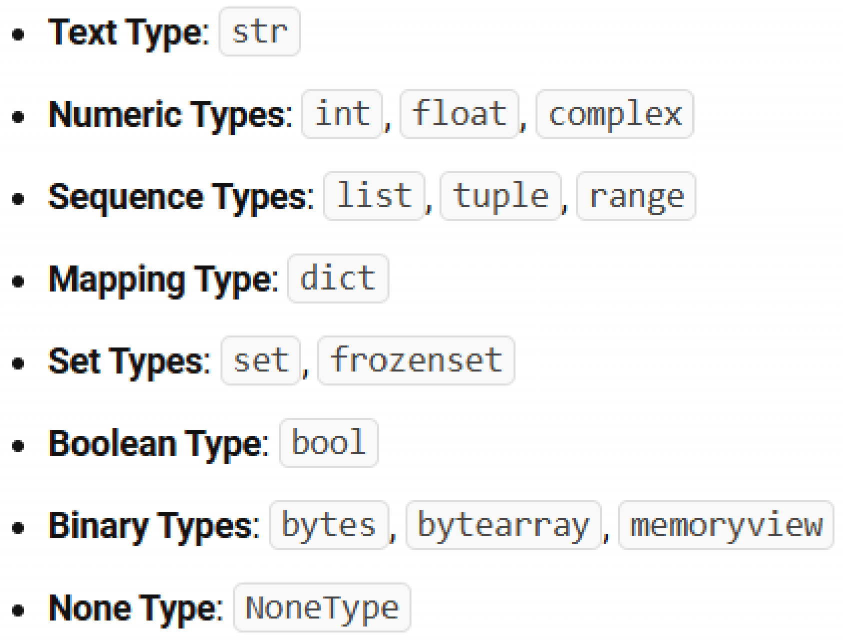
Data types

In Python, data types are the classification or categorization of data items.



Specifying Data Types

If you want to specify the data type, you can use constructor functions like :



- **Text Type**: `str`
- **Numeric Types**: `int`, `float`, `complex`
- **Sequence Types**: `list`, `tuple`, `range`
- **Mapping Type**: `dict`
- **Set Types**: `set`, `frozenset`
- **Boolean Type**: `bool`
- **Binary Types**: `bytes`, `bytearray`, `memoryview`
- **None Type**: `NoneType`

Getting and Setting Data Types

You can get the data type of any object by using the type() function.

For example, to get the data type of a variable x = 5

you would use print(type(x))

When you assign a value to a variable, the data type is set automatically.

For instance,

x = 20 would set x as an int

x = 20.5 would set x as a float

Tokens

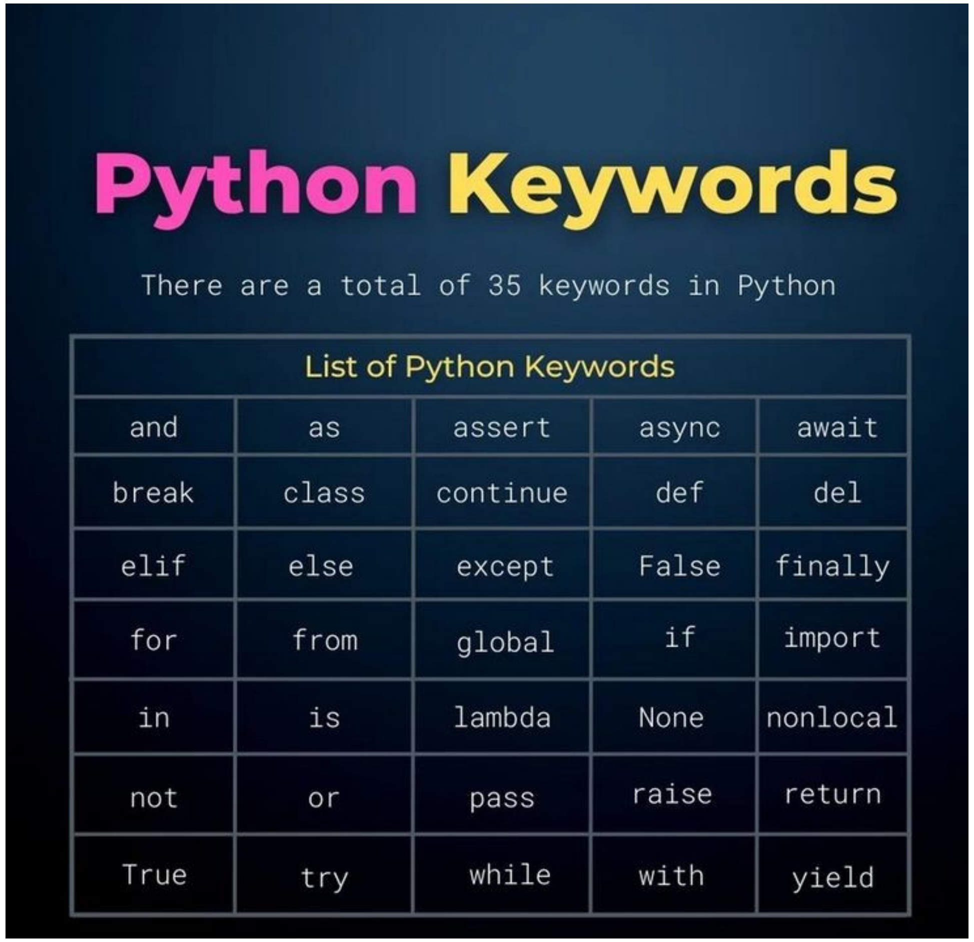
A token is the smallest individual unit in a program is known as token.

There are five types of token in python:

- Keywords
- Identifiers
- Literals
- Operators
- Punctuators

Keywords

- A python keyword is a reserved word which you can't use as a name of your variable, class, function, module and object.
- These keywords have a special meaning and they are used for special purposes in Python programming language.
- There are 35 keywords in python.
- All the keywords are in lowercase except three keywords (True, False, None).



Python Keywords

There are a total of 35 keywords in Python

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

Identifiers

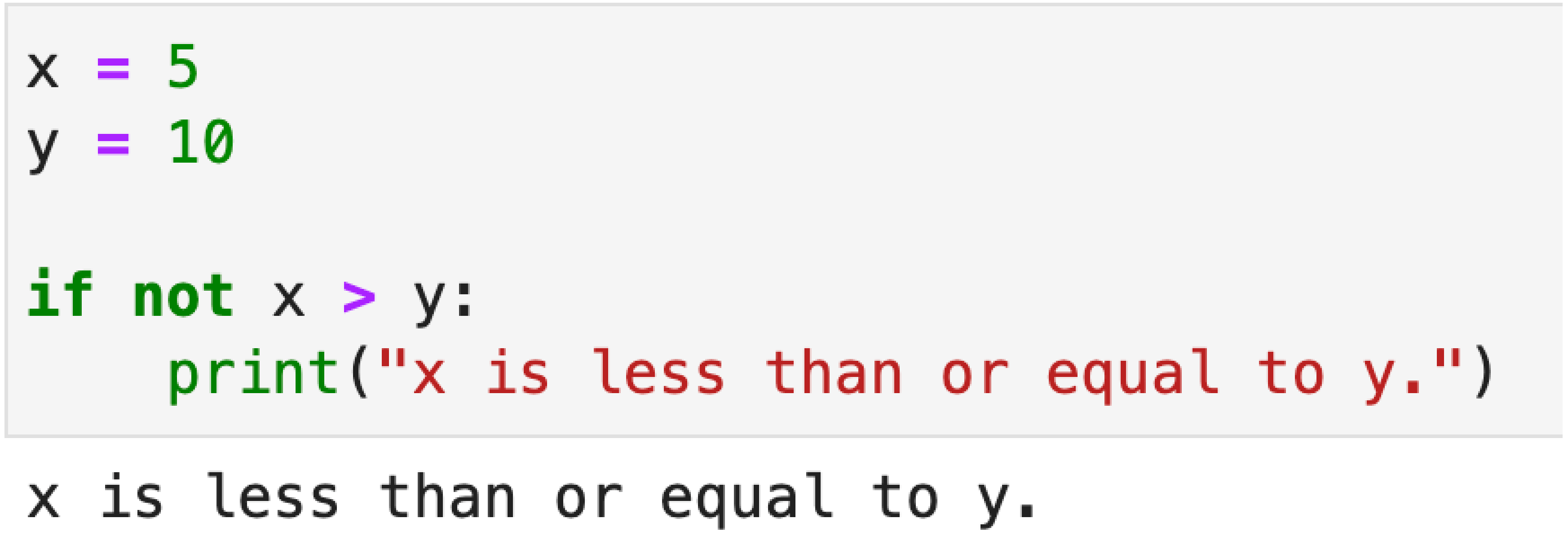
Python identifiers are user-defined names.

They are used to specify the names of variables, functions, class, module, and objects.

Rules for Variables:

- It can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore.
- It cannot start with a digit.
- Keywords cannot be used as an identifier.
- We cannot use special symbols like !, @, #, $, %, + etc. in identifier.
- _ (underscore) can be used in identifier.
- Commas or blank spaces are not allowed within an identifier.

Difference between Keywords and Variables



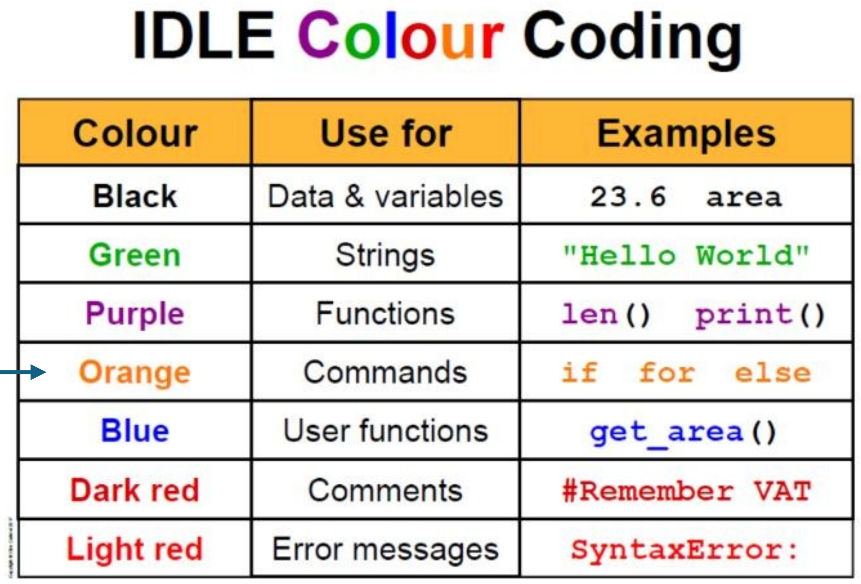
```
x = 5
y = 10

if not x > y:
    print("x is less than or equal to y.")
```

```
x is less than or equal to y.
```

IDLE Colour Coding

When you type a program in IDLE some pieces of code are shown in different colours:

- A predefined function is shown in purple.
- A string is shown in green.
- Comments are shown in red.
- If you make a mistake you may spot the colours are wrong.

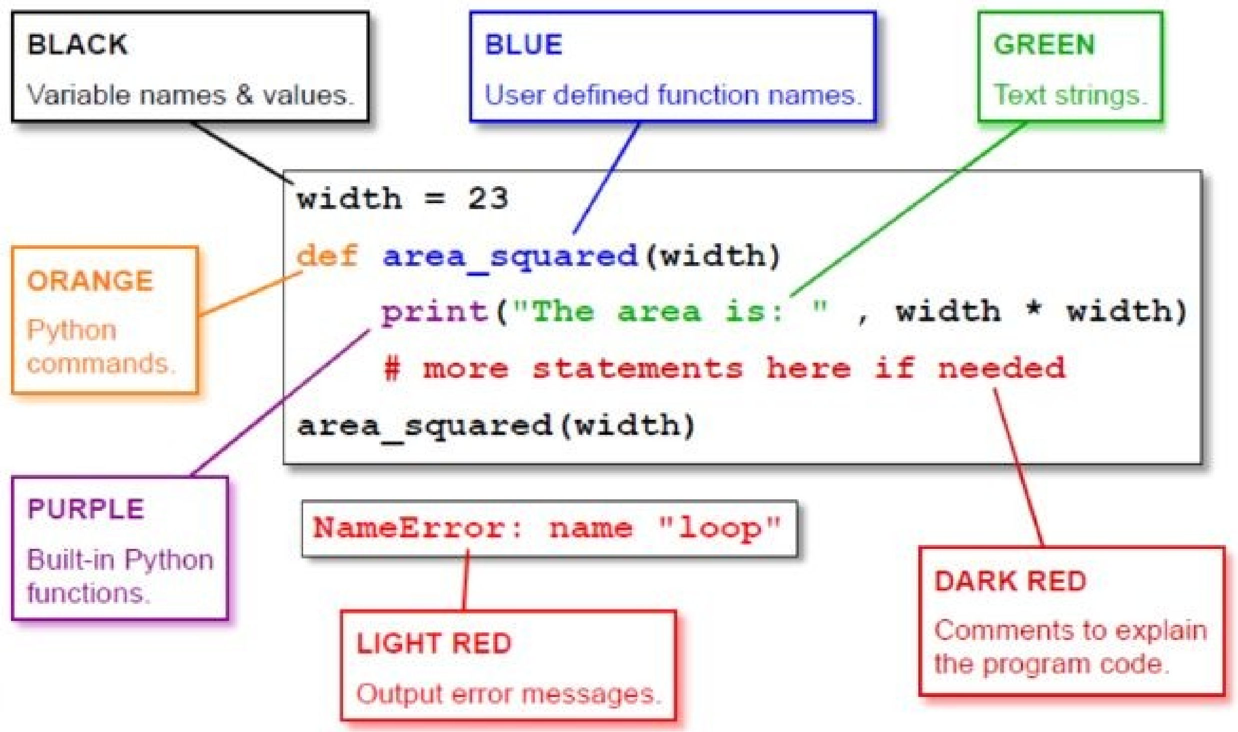


IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	`23.6 area`
Green	Strings	`"Hello World"`
Purple	Functions	`len() print()`
Orange	Commands	`if for else`
Blue	User functions	`get_area()`
Dark red	Comments	`#Remember VAT`
Light red	Error messages	`SyntaxError:`

Keywords

IDLE Colour Coding



BLACK
Variable names & values.

BLUE
User defined function names.

GREEN
Text strings.

ORANGE
Python commands.

PURPLE
Built-in Python functions.

LIGHT RED
Output error messages.

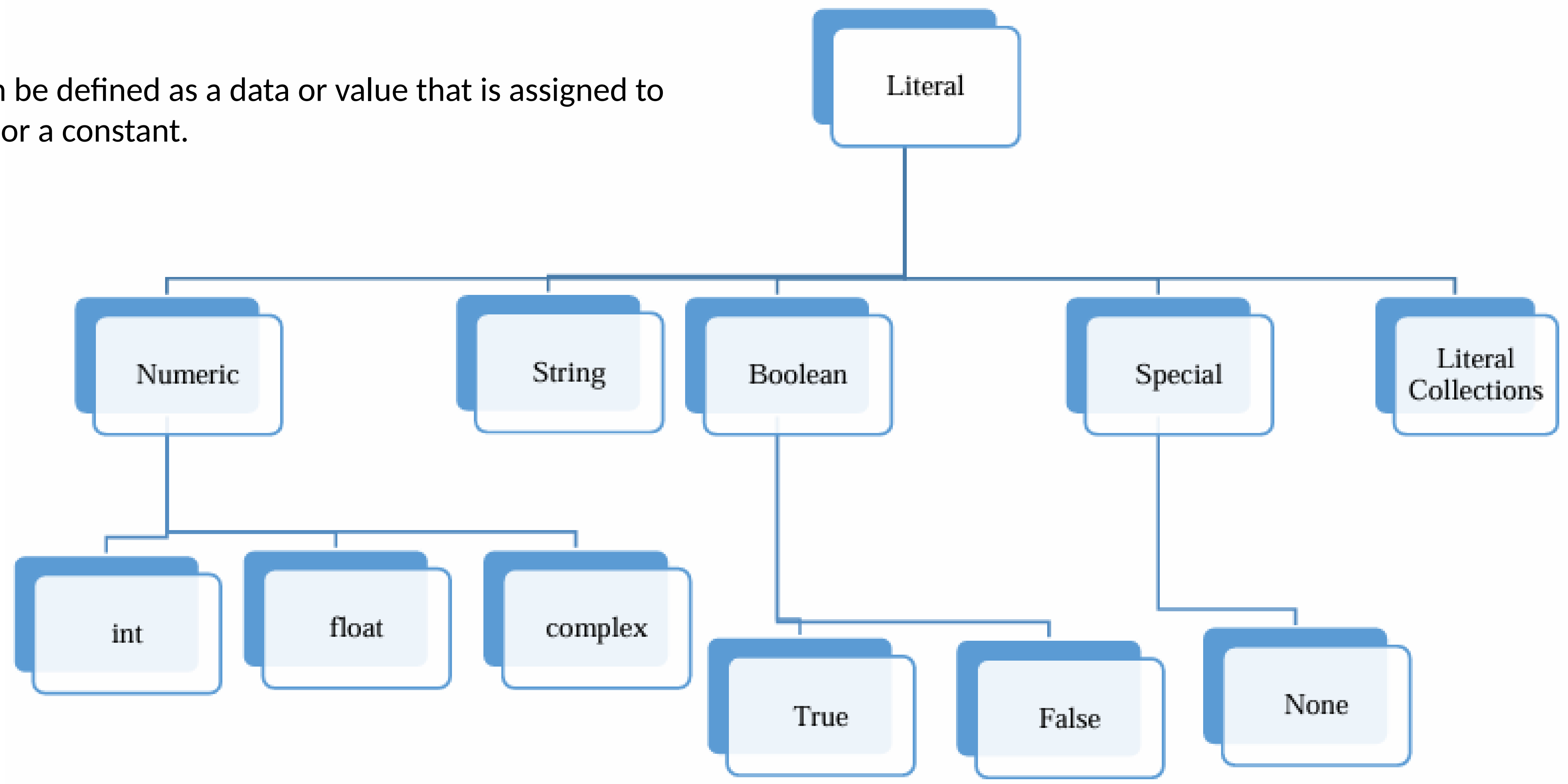
DARK RED
Comments to explain the program code.

```
width = 23
def area_squared(width)
    print("The area is: " , width * width)
    # more statements here if needed
area_squared(width)
```

```
NameError: name "loop"
```

Literals

Literals can be defined as a data or value that is assigned to a variable or a constant.



Numeric

There are three types of numeric literal:

- Integer -Both positive and negative numbers including 0.

  e.g. 4, -5, 12
- Float - These are real numbers having both integer and fractional parts.

  e.g. 5.4, -3.8
- Complex - The numerals will be in the form of a + bj, where ‘a’ is the real part and ‘b‘ is the complex part.

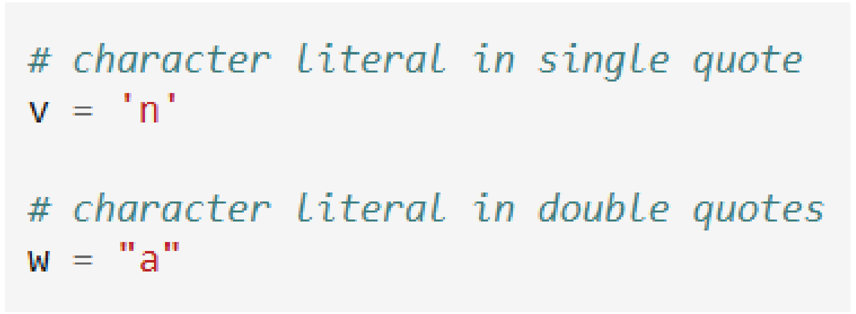
  e.g. 7+5j

Character

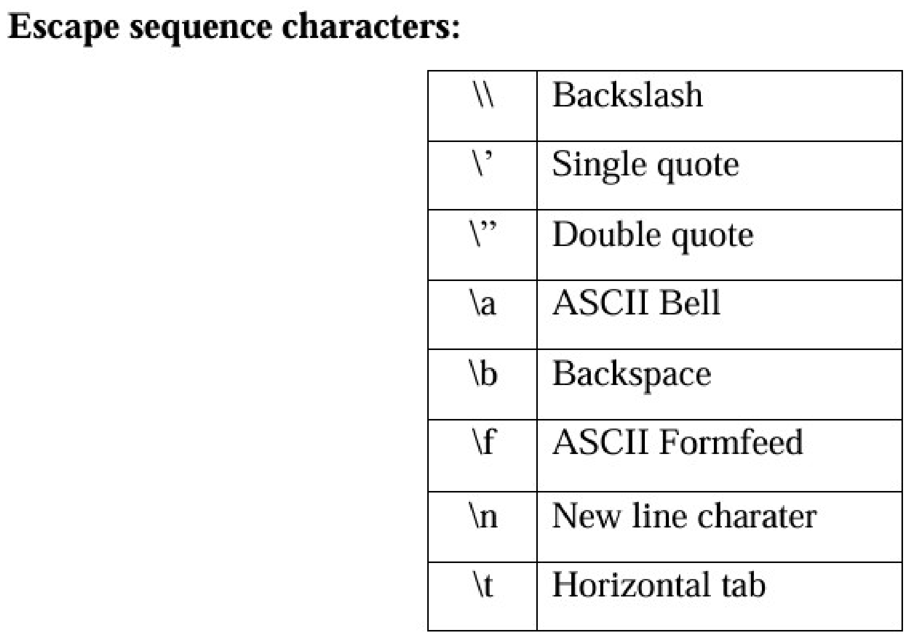
It is a set of valid characters that a language recognizes.

It is also a type of Python string literal where a single character is surrounded by single or double quotes.

Letters A-Z, a-z	Digits 0-9	Special Symbols @!$	Whitespace



```
# character literal in single quote
v = 'n'

# character literal in double quotes
w = "a"
```



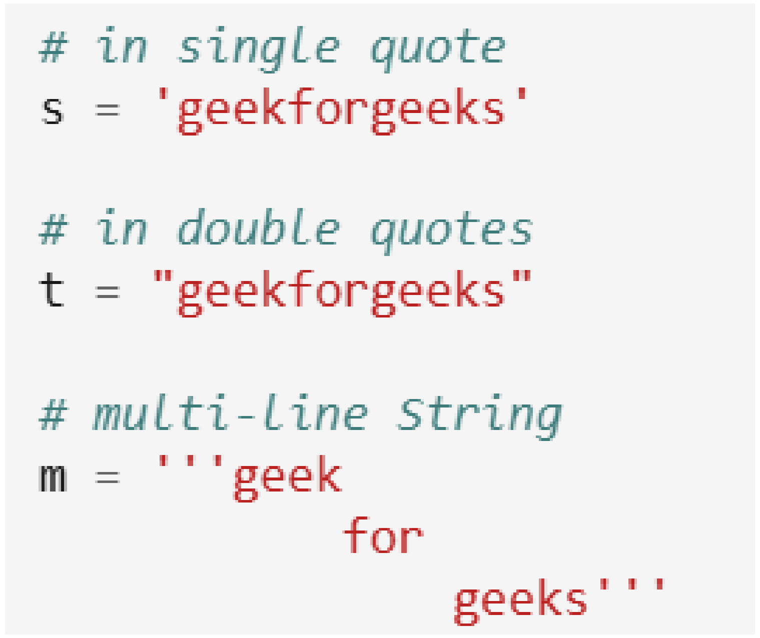
Escape sequence characters:

\\	Backslash
\’	Single quote
\”	Double quote
\a	ASCII Bell
\b	Backspace
\f	ASCII Formfeed
\n	New line charater
\t	Horizontal tab

Strings

A string is literal and can be created by writing a text (a group of characters) surrounded by a single(”), double(“), or triple quotes.

We can write multi-line strings or display them in the desired way by using triple quotes.



```
# in single quote
s = 'geekforgeeks'

# in double quotes
t = "geekforgeeks"

# multi-line String
m = '''geek
          for
               geeks'''
```

Boolean

There are only two Boolean literals in Python. They are true and false.

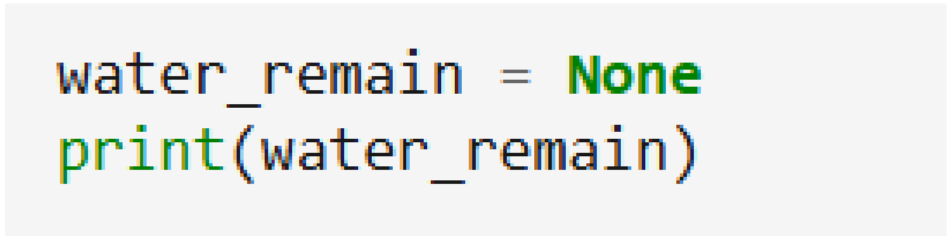
In Python, True represents the value as 1, and False represents the value as 0.

Special

Python contains one special literal (None).

‘None’ is used to define a null variable.

If ‘None’ is compared with anything else other than a ‘None’, it will return false.



```
water_remain = None
print(water_remain)
```

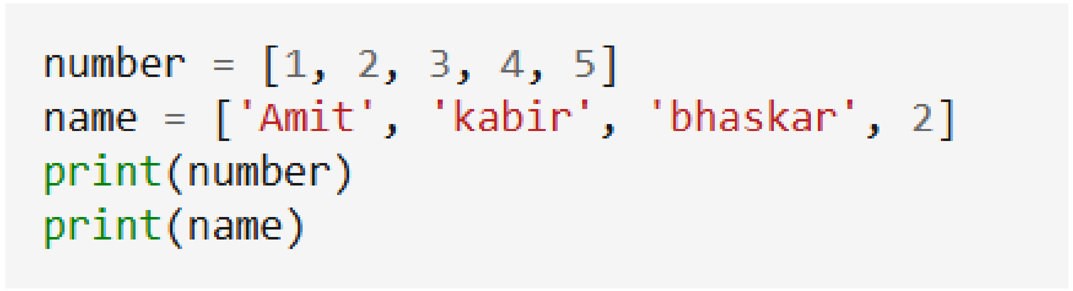
Literal Collections

Python provides four different types of literal collections:

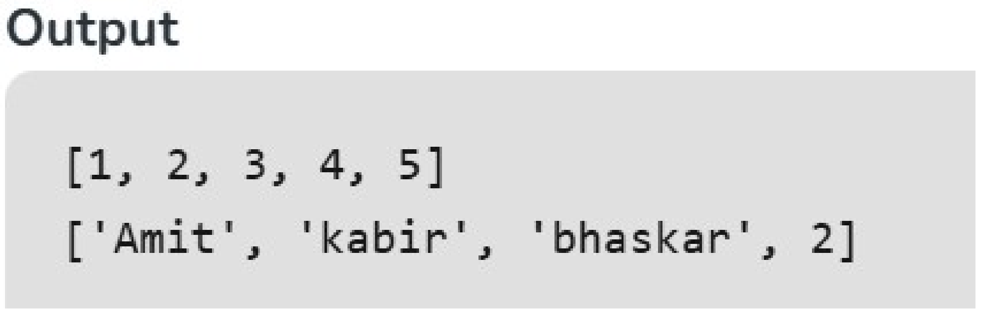
- List
- Tuple
- Dictionary
- Set

List

- The list contains items of different data types.
- The values stored in the List are separated by a comma (,) and enclosed within square brackets([]).
- We can store different types of data in a List. Hence list are heterogenous.
- Lists are mutable.



```python
number = [1, 2, 3, 4, 5]
name = ['Amit', 'kabir', 'bhaskar', 2]
print(number)
print(name)
```

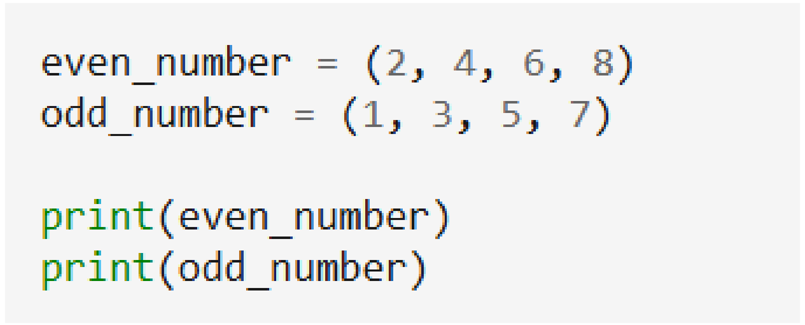


Output

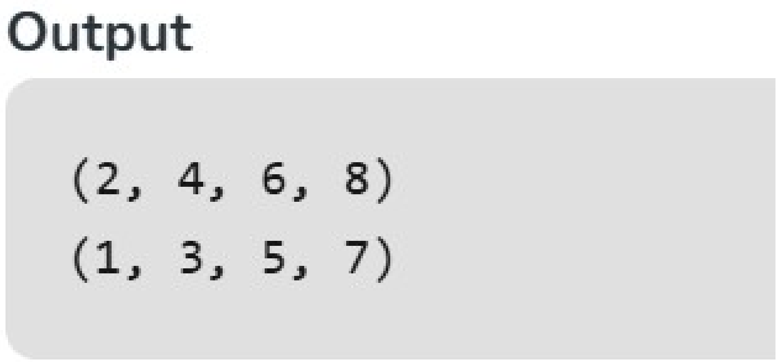
```
[1, 2, 3, 4, 5]
['Amit', 'kabir', 'bhaskar', 2]
```

Tuple

- Tuple is a collection of different data-type.
- It is enclosed by the parentheses ‘()‘ and each element is separated by the comma(,).
- Tuples are immutable.



```
even_number = (2, 4, 6, 8)
odd_number = (1, 3, 5, 7)

print(even_number)
print(odd_number)
```

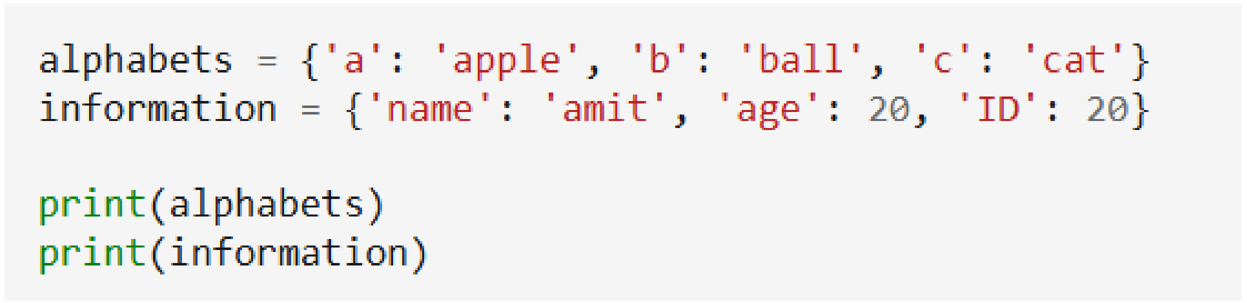


Output

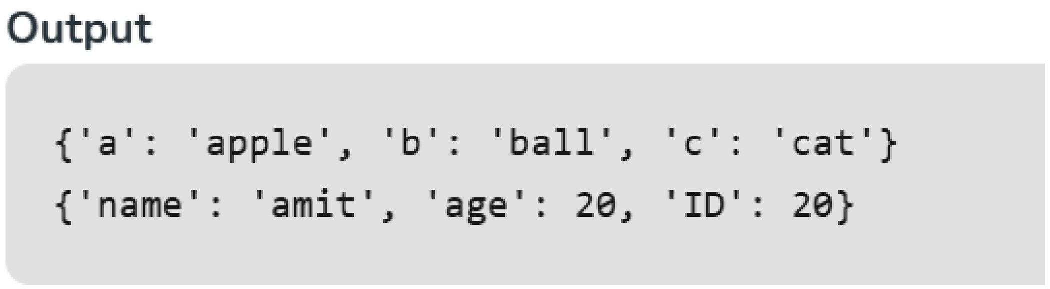
```
(2, 4, 6, 8)
(1, 3, 5, 7)
```

Dictionary

- The dictionary stores the data in the key-value pair. It is enclosed by curly braces ‘{}‘ and each pair is separated by the commas(,).
- We can store different types of data in a dictionary.
- Dictionaries are mutable.



```
alphabets = {'a': 'apple', 'b': 'ball', 'c': 'cat'}
information = {'name': 'amit', 'age': 20, 'ID': 20}

print(alphabets)
print(information)
```

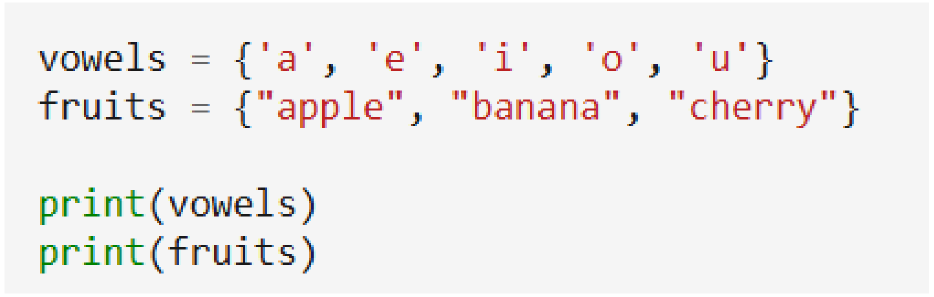


Output

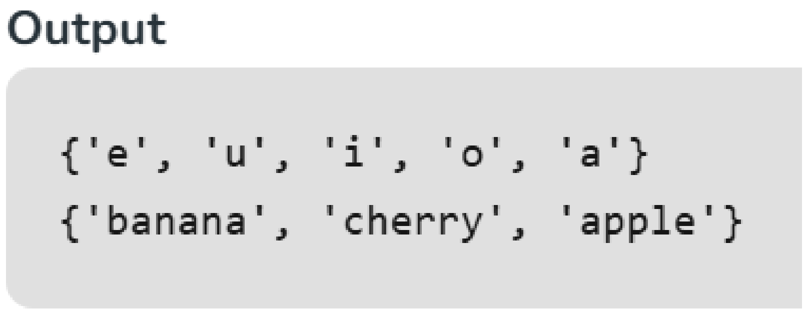
```
{'a': 'apple', 'b': 'ball', 'c': 'cat'}
{'name': 'amit', 'age': 20, 'ID': 20}
```

Set

- Set is the collection of the unordered data set.
- It is enclosed by the {} and each element is separated by the comma(,).
- Sets are mutable.

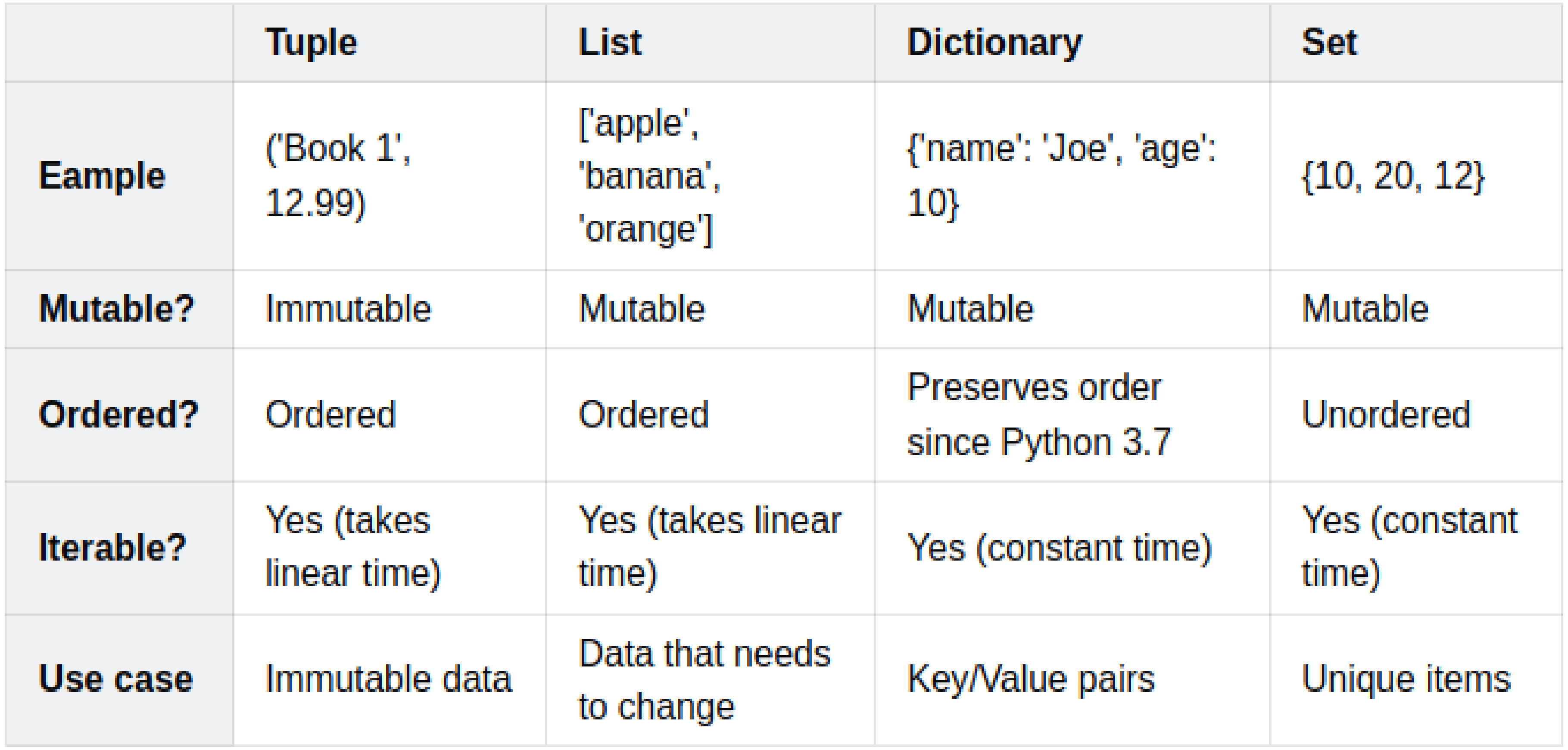


```
vowels = {'a', 'e', 'i', 'o', 'u'}
fruits = {"apple", "banana", "cherry"}

print(vowels)
print(fruits)
```



Output

```
{'e', 'u', 'i', 'o', 'a'}
{'banana', 'cherry', 'apple'}
```



	Tuple	List	Dictionary	Set
Eample	('Book 1', 12.99)	['apple', 'banana', 'orange']	{'name': 'Joe', 'age': 10}	{10, 20, 12}
Mutable?	Immutable	Mutable	Mutable	Mutable
Ordered?	Ordered	Ordered	Preserves order since Python 3.7	Unordered
Iterable?	Yes (takes linear time)	Yes (takes linear time)	Yes (constant time)	Yes (constant time)
Use case	Immutable data	Data that needs to change	Key/Value pairs	Unique items

Operators

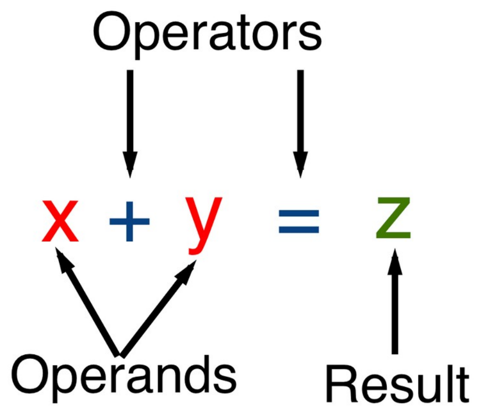
In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into different types of Python operators.

OPERATORS: These are the special symbols. Eg- + , * , /, etc.

OPERAND: It is the value on which the operator is applied.

Types of operators in python according to number of operands:

- Unary Operator
- Binary Operator



Unary Operator

- Performs the operation on one operand, meaning it affects only one value or variable.
- Unary operators are commonly used in programming languages to perform various operations such as changing the sign of a value, incrementing or decrementing a value, or performing logical negation.
- May appear before or after the operand.

Examples of Unary Operators:

Unary minus (negative number, eg :-5)

Unary plus (positive number, eg: 5)

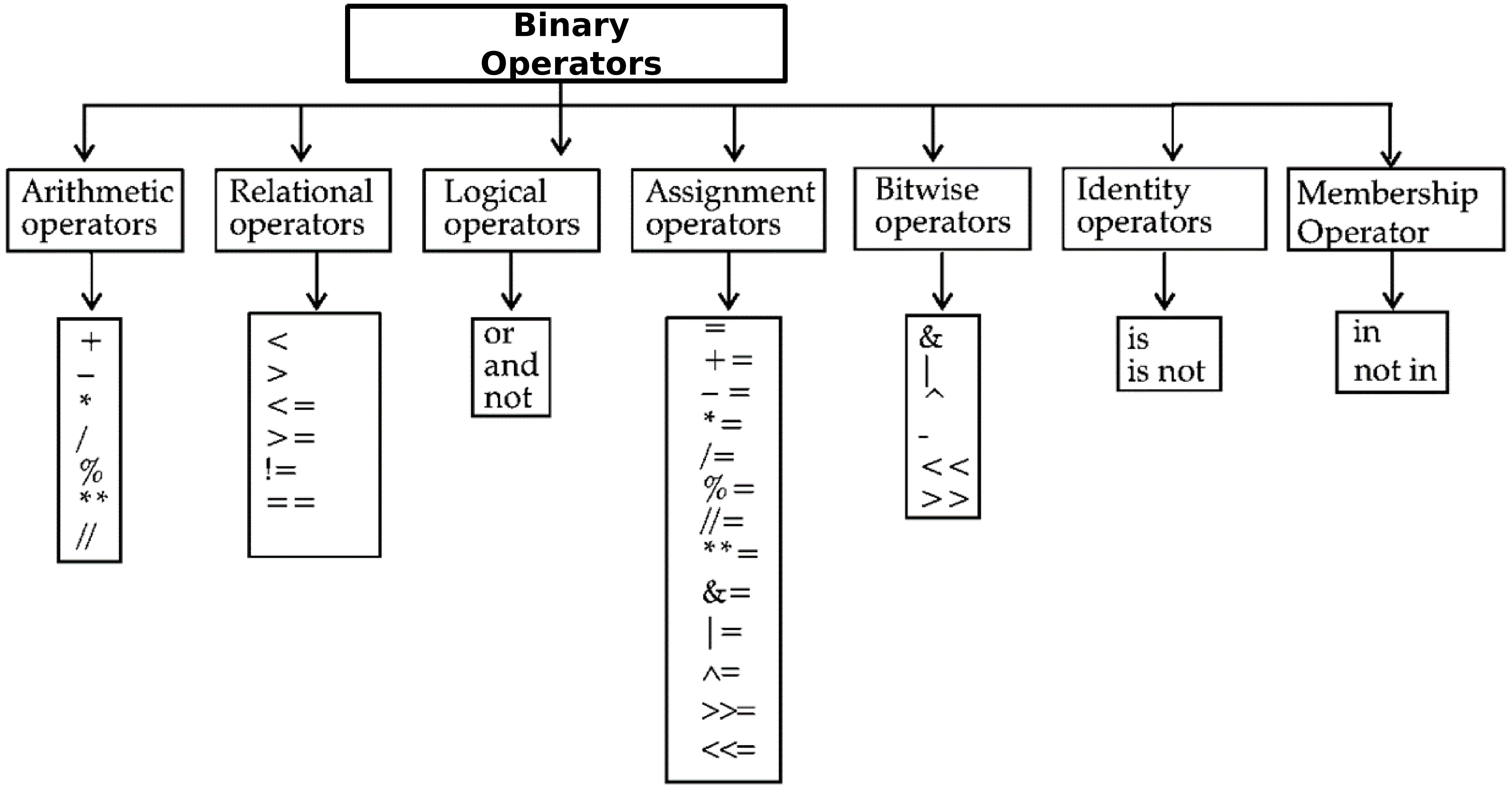
Increment (++)

Decrement (- -)

Logical NOT (!x)

Binary Operator

- Binary operators are commonly used in programming languages to perform various operations such as arithmetic operations, logical operations, and bitwise operations.
- Appears between the operands.

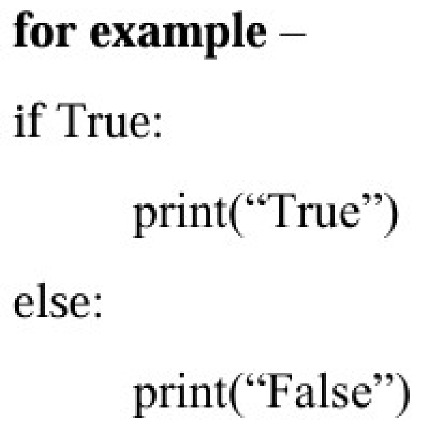

Binary Operators
Arithmetic operators
+
–
*
/
%
**
//
Relational operators
<
>
<=
>=
!=
==
Logical operators
or
and
not
Assignment operators
=
+=
–=
*=
/=
%=
//=
**=
&=
|=
^=
>>=
<<=
Bitwise operators
&
|
^
-
<<
>>
Identity operators
is
is not
Membership Operator
in
not in

Basic terms of a Python Programs

- Blocks and Indentation
- Statements
- Expressions
- Comments

Blocks and Indentation

- Python uses indentation to define blocks of code, unlike other languages that use braces.
- This means that the whitespace at the beginning of the line is significant and must be consistent.
- 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.



for example –

```
if True:
        print(“True”)
else:
        print(“False”)
```

Statements

A line which has the instructions or expressions.

This could be anything written in Python, such as a command to print something to the console, a variable assignment, or a control flow statement like if, for, or while.

Python statements can be broadly categorized into simple and compound statements:

Simple Statements: These are the most basic kind of statement that does something. These are single line statements. Examples include expression statements, assignment statements, and the pass, del, return, import, continue, and break statements.

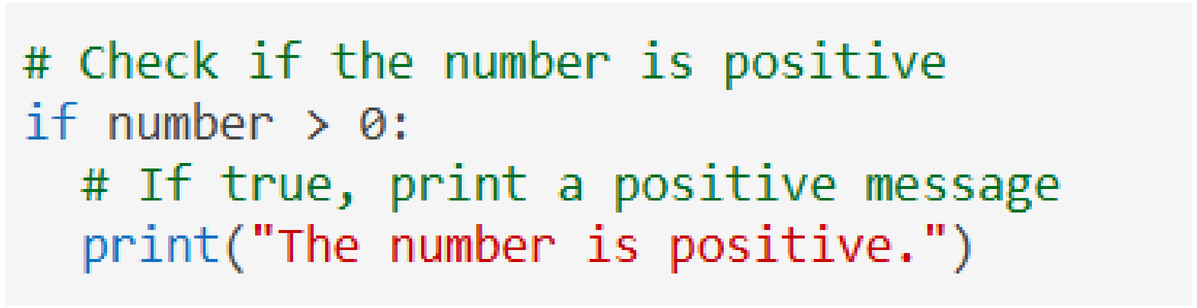
Compound Statements: These contain groups of other statements and control the execution flow of the program. Examples include if, while, for, try, and with statements.

Comments

- Comments are not executed.
- Comments in Python start with the # symbol and are used to explain what the code is doing.
- All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.
- They're crucial for making your code understandable to others and to your future self.
- There are two types of comments in python:

Single line comment

Multi-line comment



```
# Check if the number is positive
if number > 0:
  # If true, print a positive message
  print("The number is positive.")
```

Comments Contd..

Single line comment: This type of comments start in a line and when a line ends, it is automatically ends.

Single line comment starts with # symbol.

Example: if a>b: # Relational operator compare two values

Multi-Line comment: Multiline comments can be written in more than one lines. Triple quoted ‘ ’ ’ or “ ” ”) multi-line comments may be used in python. It is also known as docstring.

Example:

‘’’ This program will calculate the average of 10 values.

First find the sum of 10 values

and divide the sum by number of values

‘’’

Punctuators

- Punctuators in Python are symbols used to structure and organize code.
- They are used to indicate the beginning and end of statements, define data types, and control the flow of execution.
- Some of the commonly used punctuators in Python are:

Parentheses: () - Used to group expressions and to call functions.

Brackets: [] - Used to define lists and access elements in a list.

Curly Braces: {} - Used to define dictionaries and sets.

Colon: : - Used to indicate the beginning of a block of code, such as in loops, functions, and conditional statements.

Comma: , - Used to separate items in a list, tuple, or dictionary.

Expressions

An expression is a piece of computer code that represents a value.

Expressions in Python are constructs made up of variables, operations, and values that a Python interpreter can evaluate to produce another value.

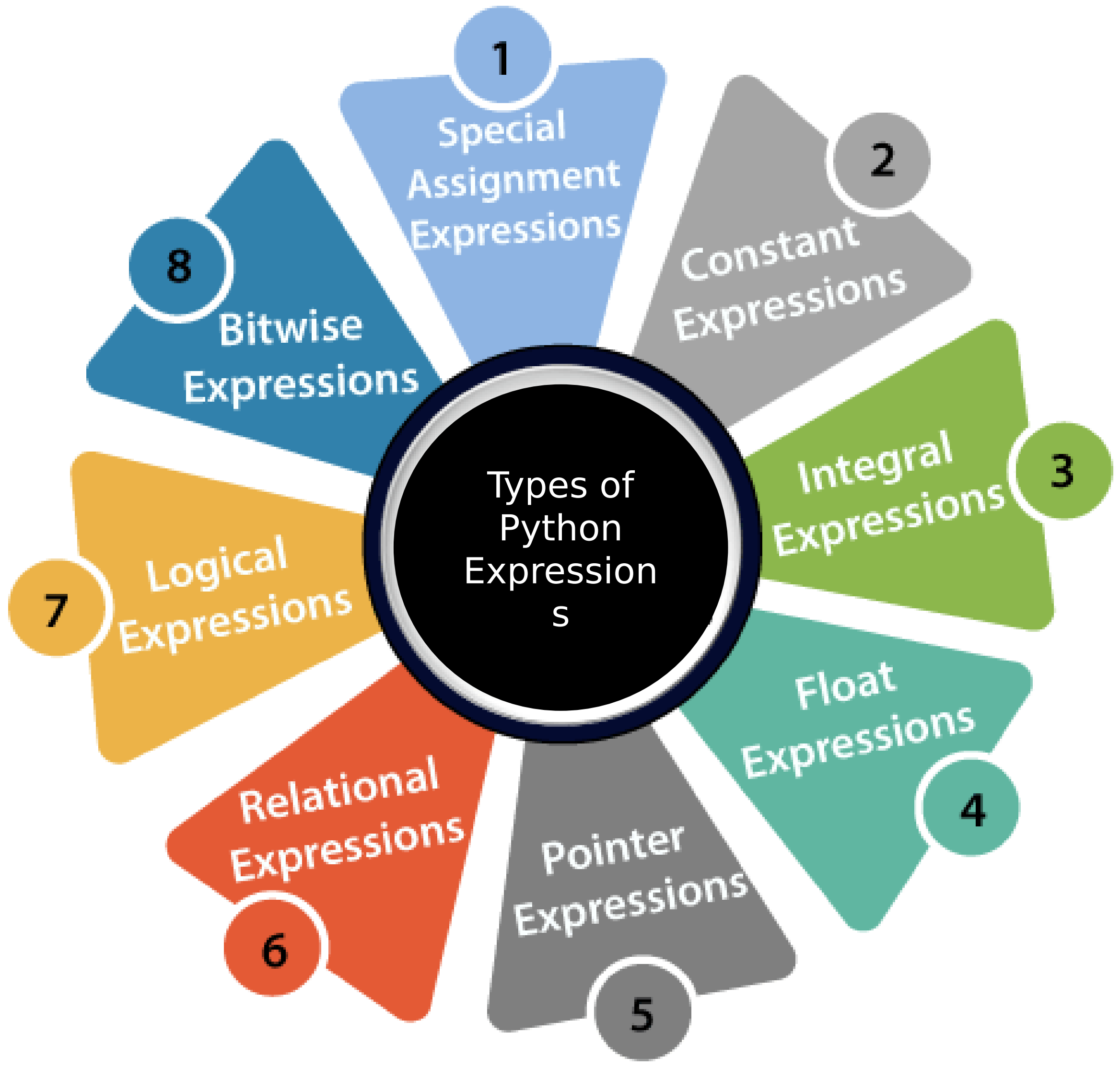
They are an integral part of programming in Python, allowing for the execution of operations and the storage of resulting values.

Examples:

a + b

(a+b) AND (c-d)

Types of Expressions


Types of Python Expressions
1
Special Assignment Expressions
2
Constant Expressions
3
Integral Expressions
4
Float Expressions
5
Pointer Expressions
6
Relational Expressions
7
Logical Expressions
8
Bitwise Expressions