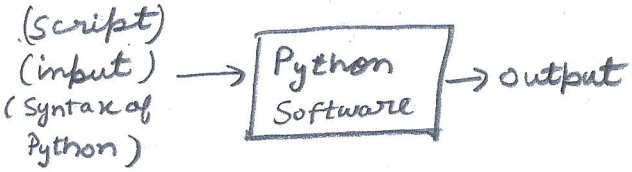
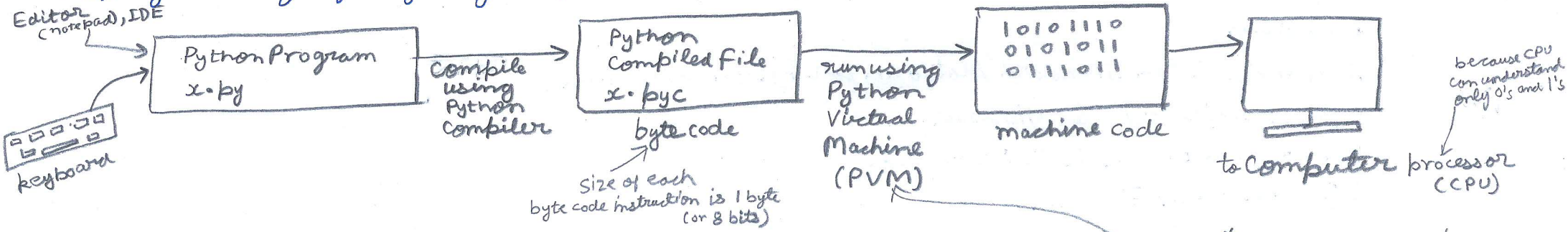



| | |
|---|---|
| I | **Introduction:** The Programming Cycle for Python , Python IDE, Interacting with Python Programs , Elements of Python, Type Conversion.<br>**Basics:** Expressions, Assignment Statement, Arithmetic Operators, Operator Precedence, Boolean Expression. |

## The programming cycle for Python




## Steps of execution of a Python Program

convert* the byte code into machine code and sends that machine code to the computer processor (CPU) for execution. PVM is equipped with an interpreter*. Since PVM is playing the main role, often the Python Virtual machine is also called an interpreter.

Python IDE: Integrated Development Environment makes writing programs fast (less time consuming), easy. This happens because several features are pre-installed inside IDE.

eg. IDLE is Python's Integrated Development and Learning Environment,

PyCharm,

Anaconda is the standard platform for Python data science,

Visual Studio Code by Microsoft,

PyDev is a Python IDE for Eclipse,

Wing Python IDE etc.

## Interacting with Python Programs: Two Modes:

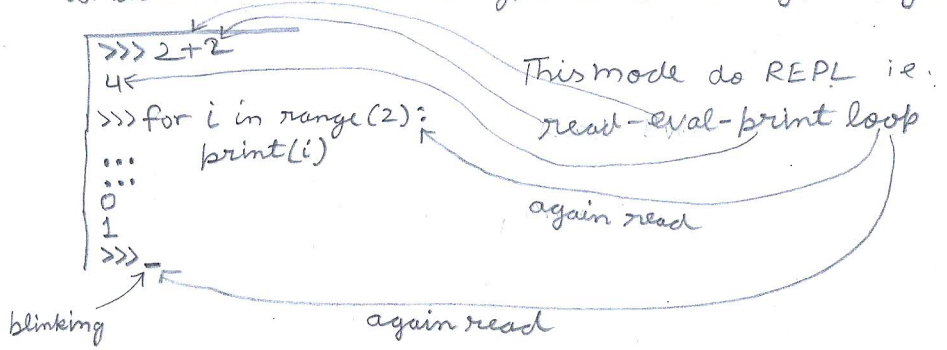
Python programs are also known as scripts.

### Interactive Mode Programming

In this, we invoke the interpreter without passing any script or Python file.

We can start the Python command line interpreter or the Python shell in IDLE and start passing instructions and get instant results.

Windows + R button → type cmd → type python




### Script Mode Programming

Saving code in a Python file (extension .py). When the script is executed the interpreter is invoked and it is active as long as the script is running. Once all the instructions of the script are executed the interpreter is no longer active.

a.py

```
a=10
b=20
c=a+b
print("Sum is", c)
```

Windows + R button → type cmd

```
C:\users\Folder1> python a.py
Sum is 30

C:\users\Folder1>
```

blinking

## Elements of Python:

In English language we have words which make a sentence. eg noun, verb etc.
In Python language we have words in the form of Identifiers.
Identifiers can be variables, datatypes, functions, class, Keywords, packages etc.
This means these words are classified according to their usage.
Therefore, identifier is smallest identifying unit in the program.

We tell meaning to the Python interpreter

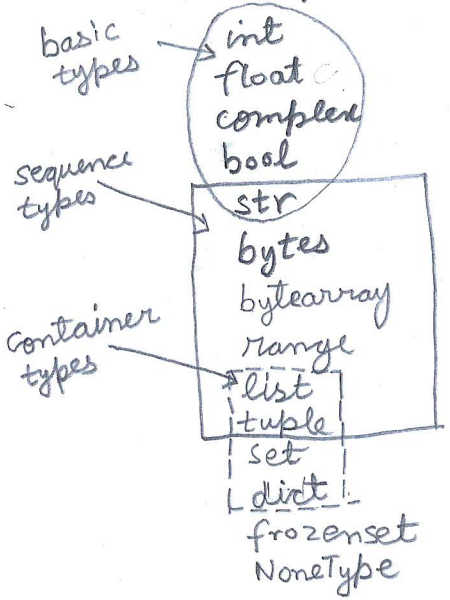
predefined words (meaning already known to Python interpreter)
(dont use any keyword as name of variable)

Operators
2 + 2 — values } expression

## Type Conversion:

Type means data type. In short we say data type as type.
14 data types are listed below. In Python, type is decided dynamically.

List of built-in data types




what is need of type conversion?

Python defines datatype conversion functions to directly convert one data type to another.

```
>>> x = '123'
>>> y = 6
>>> int(x) + y
129
>>>
```

eg. x = "123"
y = 5
now x + y
then what?

type(x) → <class 'str'>
type(y) → <class 'int'>
X (not allowed)

str ⟶ int( )

i.e we converted a "integer numeric string" to integer by passing "123" in int() function

Like this we can covert one type to another type

## Conversion functions

int() float() str() complex()
list() tuple() set() dict ord()

If we want to convert bool to int then

bool ⟶ int()

now bool has 2 values ie. True False. So we can use any of these 2 values as an argument to int() function

ie. >>>int(True)
1
>>>int(False)
0

ie we converted boolean to integer

int ⟶ bool()

Non-zero
Zero

this means we have to supply int values inside bool() function

Zero value for int is 0

Non-zero value for int is all integers other than 0

eg >>>bool(0) → False → bool type

>>>bool(2) → True → bool type

>>>bool(-3) → True → bool type

### False Data Type Values (or zero values)

| Data Type | Value |
|---|---|
| Integer | 0 |
| Float | 0.0 |
| String | '' (empty string) |
| List | [] |
| Tuple | () |
| Dictionary | {} |
| Complex | 0j |

Basics: Expressions always Evaluate down (reduce down) to a single value.

Consists of values and operators

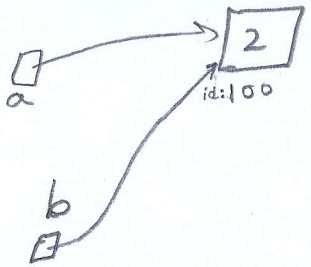
>>> 2+2
4

>>> 2 ← expression 2 evaluates to itself.
2

2+2 is expression because it eval down to a single value 4.

Assignment Statement : The symbol = is used as assignment operator

>>> a = 2 means integer object 2 is created in memory. Then a is a reference variable which is referring 2. This happens because a has the identity where 2 is located in memory




2 is located in location 100. Now a has 100, then a will only refer to 2.

>>> b = a ← then b will start referring to what a is referring (now b has also 100)

# Arithmetic operators

a = 13
b = 5

| | | | |
|---|---|---|---|
| ** | Exponent | a**b | 371293 |
| // | Floor division | a//b | 2 |
| % | Modulus | a%b | 3 |
| / | division | a/b | 2.6 |
| + | addition | a+b | 18 |
| - | subtraction | a-b | 8 |
| * | multiplication | a*b | 65 |

5//2 → 2  -5//2 → (-3) ← minimum integer close to -5/2

gives remainder of division

**Operator Precedence:** An expression or formula may contain several operators. In such a case, the programmer should know which operator is executed first and which one is executed next.

The sequence of execution of the operators is called operator precedence.

Precedence of Operators in Python

| | Operator | Name |
|---|---|---|
| HIGHEST → | () | Parenthesis |
| | ** | Exponentiation |
| | -, ~ | Unary minus, Bitwise complement |
| | *, /, //, % | Multiplication, Division, Floor division, Modulus |
| | +, - | Addition, Subtraction |
| | <<, >> | Bitwise left shift, Bitwise right shift |
| | & | Bitwise AND |
| | ^ | Bitwise XOR |
| | \| | Bitwise OR |
| | >, >=, <, <=, ==, != | Relational (comparison) operators |
| | =, +=, -=, *=, **=, /=, //=, \|=, &=, ^=, %=, <<=, >>= | Assignment operators |
| | is, is not | Identity operators |
| | in, not in | Membership operators |
| | not | Logical not |
| | or | Logical or |
| LOWEST → | and | Logical and |

these 12 are also known as compound Assignment operators, made by using 2 symbols eg: a=1

a += 6+3 → a = a+9 → a = 10

Note: + has higher priority than =

b = 5 b *= 4 → b = b*4 → b = 5*4 → b = 20

Precedence represents the priority level of the operator. The operators with higher precedence will be executed first than of lower precedence.

Associativity is the order in which an expression is evaluated that has multiple operators of the same precedence. Almost all the operators have left-to-right associativity in Python.

** has right-to-left associativity

eg. >>> 2**3**2
512

L ← R

i.e. first $3^2$ is evaluated which is 9, then $2^9$ is evaluated which is 512. This happens Right-to-Left.

abhi.c

**Boolean Expression (or Logical Expression)** : is an expression that evaluates down to a Boolean value, i.e. either True or False.

**Relational (Comparison)** operators are used to compare values and evaluate down to a single Boolean value of either True or False.

| Operator | meaning |
|---|---|
| == | Equal to |
| != | not equal to |
| < | less than |
| > | greater than |
| <= | less than or equal to |
| >= | greater than or equal to |

compares value, by seeing CONTENT and TYPE both sides
(these both must be same in order to get True as result)
eg 5 (int) 8 (int)

(Exception case: >>> 5 == 5.0 True)

```
>>> x = 5
>>> y = 8
>>> x == y
False
>>> x != y
True
>>> x < y
True
>>> x > y
False
>>> x <= y
True
>>> x >= y
False
```

**Logical Operators:** The 3 logical operators are used to compare values. They evaluate expressions down to Boolean values, returning either True or False.

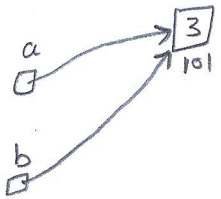
| Operator | meaning | |
|---|---|---|
| and | True if both are true | x and y |
| or | True if atleast one is true | x or y |
| not | True only if false | not x |

```
>>> (9 > 7) and (2 < 4)
True
>>> (9 == 9) or (6 != 6)
True
>>> not (3 <= 1)
True
```

**Identity operator** Compares identity both sides.

| Operator |
|---|
| is |
| is not |

```
>>> a = 3
>>> b = a
>>> a is b
True
>>> b is a
True
```




a and b have 101 as identity therefore result is True.

**Membership operator**: returns True is a sequence with the specified value is present in the object.

| Operator |
|---|
| in |
| not in |

```
>>> x = "abc"
>>> "a" in x
True
>>> "A" in x
False
```