Installation instructions:

-Linux: Python comes preinstalled with Linux, so the procedure should be simple.

Follow the instructions at this link to update to the latest version:
https://docs.python-guide.org/starting/install3/linux/

At the end, you can also choose to use an alias, which makes things easier.
Enter the following command in your terminal: alias python=python3

-Mac: Macs come preinstalled with Python 2.7, which sadly, will reach the end of it's life on

01/01/2020.
Follow these instructions at this link to install Python3:
https://programwithus.com/learn-to-code/install-python3-mac/

-Windows: Installing Python on Windows, while easy, is a long process.

Follow these instructions at this link to install Python3:
https://realpython.com/installing-python/#windows

After installing Python, you're going to need to install pip, to be able to install libraries without having to always clone the github repo.

-Linux: Open your terminal and type: sudo apt install python3-pip

-Mac: If you followed the instructions above, pip should be installed already.

-Windows: If you followed the instructions above, pip should be installed already.

To check if pip is installed, open terminal/command prompt and type: pip --version
On Mac, type pip3 --version
If any doubts go to- https://stackoverflow.com/questions/4750806/how-do-i-install-pip-on-windows

Python Fundamentals

Python character Set

- lower ⇐ a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
- upper ⇐ A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
- digit ⇐ 0|1|2|3|4|5|6|7|8|9
- ordinary ⇐ |(|)| [|] | { | } |+|-|*|/|%|!|&| | |~|^||,|.|:|;|$|?|#
- graphic ⇐ lower | upper | digit | ordinary
- special ⇐ ’ | " | \
- white space ⇐ | → | ←- (space, tab, or newline)

Python Program Structure

Python program contains various components like : -

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

Expressions

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

e.g., a = 15

Statements

- 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.

e.g.

a=10

print(a)

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”)
```

Functions

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

```
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)
```

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); block
       print("Prints only if condition is TRUE");
       else: block
       print(food); block
       print("Prints only if condition is FALSE");
print(sit);
menu(drink)
```

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 un-indented 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 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 1:
 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);
```

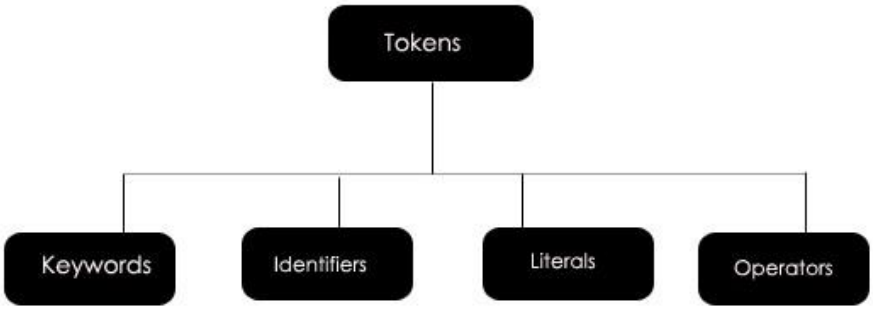
We can also write the above program as

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

Tokens or Lexical Unit

The smallest individual unit in a program is known as a Token or a lexical unit.

Python has following tokens:



Keywords

Keywords are words that convey a special meaning to the language compiler/interpreter. These are reserved for purpose and must not be used as normal identifier names.

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 special	
break	except	in	raise	

The above keywords may get altered in different versions of Python. Some extra might get added or some might be removed. You can always get the list of keywords in your current version by typing the following in the prompt.

```
>>> import keyword
>>> print(keyword.kwlist)
```

Identifier

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

Rules for writing identifiers

2. 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_1` and `print_this_to_screen`, all are valid example.
3. An identifier cannot start with a digit. `1variable` is invalid, but `variable1` is perfectly fine.
4. Keywords cannot be used as identifiers.

```
>>> global = 1
File "<interactive input>", line 1
global = 1
^
SyntaxError: invalid syntax
```

5. We cannot use special symbols like **!**, @, #, **$**, **%** etc. in our identifier.

```
>>> a@ = 0
  File "<interactive input>", line 1
    a@ = 0
     ^
SyntaxError: invalid syntax
```

6. Identifier can be of any length.

Literals

Literals are data items that have a fixed value

String Literals

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

Eg:

"Aman" , '12345'

Types of Strings:

There are two types of Strings supported in Python:

a) Single line String- Strings that are terminated within a single line are known as Single line Strings.

```
>>> text1='hello'
```

b) Multi line String- A piece of text that is spread along multiple lines is known as Multiple line String.

There are two ways to create Multiline Strings:

1). Adding black slash at the end of each line.

```
>>> text1='hello\
user'
>>> text1
'hellouser'
>>>
```

2).Using triple quotation marks:-

```
>>> str2='''welcome
to
SSSIT'''
>>> print (str2)
welcome
to
SSSIT
>>>
```

Numeric Literals

Numeric Literals are immutable. Numeric literals can belong to following four different numerical types.

Int(signed integers)	Long(long integers)	float(floating point)	Complex(complex)
Numbers(can be both positive and negative) with no fractional part.eg: 100	Integers of unlimited size followed by lowercase or uppercase L eg: 87032845L	Real numbers with both integer and fractional part eg: -26.2	In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. eg: 3.14j

Boolean Literals

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

Special Literals

Python contains one special literal i.e., None.

None is used to specify to that field that is not created. It is also used for end of lists in Python.

Operators

Types of Operator

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then –

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) –	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.
Assume variable a holds 10 and variable b holds 20, then −

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then −

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a

*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows –

a = 0011 1100
b = 0000 1101
---
a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a = 1100 0011

There are following Bitwise operators supported by Python language

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)

~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Used to reverse the logical state of its operand.

Python Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below −

Operator	Description	Example
Is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

Sr.No.	Operator & Description
1	****** Exponentiation (raise to the power)
2	**~ + -** Complement, unary plus and minus (method names for the last two are +@ and -@)
3	*** / % //** Multiply, divide, modulo and floor division
4	**+ -** Addition and subtraction
5	**>> <<** Right and left bitwise shift
6	**&** Bitwise 'AND'
7	**^ |** Bitwise exclusive `OR' and regular `OR'
8	**<= < > >=** Comparison operators
9	**<> == !=** Equality operators
10	**= %= /= //= -= += *= **=** Assignment operators

11	**is is not** Identity operators
12	**in not in** Membership operators
13	**not or and** Logical operators

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- ' " # \ ( ) [ ] { } @ , : . =