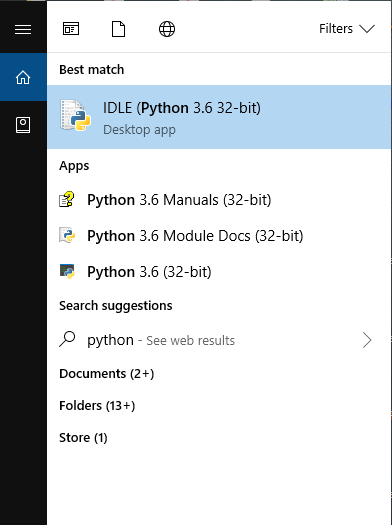
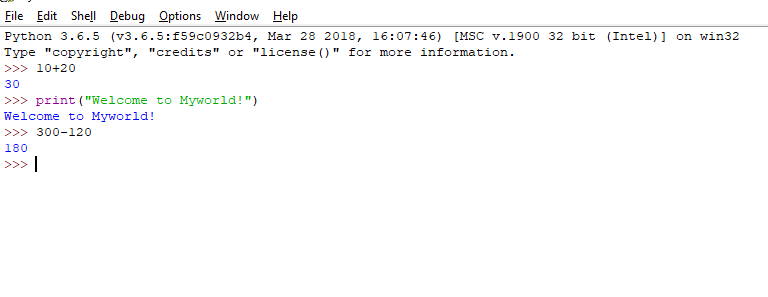
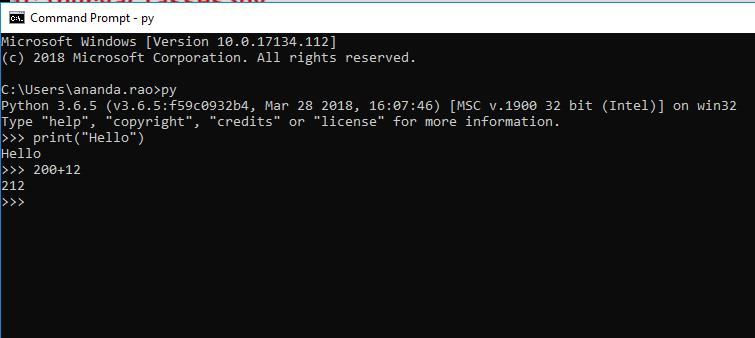
|  |  |
| --- | --- |
| Python 2 | Python 3 |
| Support till 2020 only | * Not upgraded version of Python 2 but completely new * No guarantee on backward compatibility |

**REPL Tool: Read, Evaluate, Print, Loop**



**REPL Editor:**





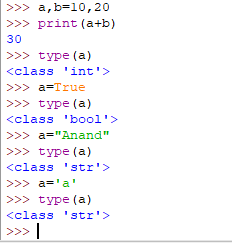
**Language Features:**

* Python is general purpose high level language [*directly communicates with machine; can be understood by programmer like Java, C#*]
* Guido Van Rossum, 1989 [National Research Institute, Netherlands], Officially made available to public on Feb 20th , 1991



<https://en.wikipedia.org/wiki/Guido_van_Rossum>

* Dynamically Typed Programming (C, Java is Static Typed )



* Name because of popular BBC TV shows called Monty Python’s Circus 1969-74
* Features borrowed from other language
* Functional programming from **C**
* OOP from **C++** (not from Java since Java was not present at that time, 1995)
* Scripting language features from **Perl & Shell** script
* Modular programming features from **Modula-3** [*programs are divided into modules*]
* Syntax borrowed from C language & ABC language
* Where we can use python?
* Desktop applications
* Web applications [Django, Flask, Pyramid frameworks provides]
* Database applications
* Networking applications
* Games
* Data Analytics
* Machine learning
* AI
* IOT
* Features of Python
* 30 keywords in Python [53 in Java]; more readability & less coding effort
* Free ware and Open source [Free ware; no licensing cost] [You can able to see the source code and customize as per your requirement and can implement you can release your version]
* High level programming language
* Platform independent [write once and run anywhere]
* Portability [migration from one machine to another machine without any changes]
* **Dynamically Typed programming** [no need to declare type for variable]
* Support for both Procedure oriented and Object oriented [procedure oriented: without class we can manage with global variables and functions][***In java lambda expressions are introduced in version 8 to support procedure oriented***]
* Interpreted programming language
* **Python is extensible** [***we can use other programming languages code in python***]
* *Native language support is there in python*
* *We can improve the performance of application by opting other language programs but at the cost of platform independence*
* Embedded [*Python code can be embedded in other language codes (inter-operability)*]
* Rich library support is present in python [***Extensive Library***]
* Limitations of python
* Performance is not up to the mark since it is interpreted language
* Not suited for mobile application development
* Flavours of Python
* CPython [*standard python and can work with apps developed on C*]
* Jython or JPython
* IronPython [C#]
* Pypy [***Python for Speed****, inside Python VM, JIT compiler will be there*]
* RubyPython
* AnacondaPython [***To handle large data sets (Big Data)*]**
* Stackless [***Python for concurrency***]
* Python Versions
* Version 1.0 in 1994
* Version 2.0 in October 2000
* Version 3.0 in December 2008
* Version 3.6.3 in 2016
* No backward compatibility
* **Identifiers**
* A name in python program is called Identifier
* It can be variable name or function name or class name …etc.
* Rules to define Identifiers:

1. Alphabet symbols (both upper & lower case), Digits (0-9), underscore (\_)
2. Identifiers should not start with digit
3. Python identifiers are case sensitive [total =10 & TOTAL=100 are not same ]
4. Keywords are not allowed to use as identifiers [x=10 & def=20]
5. There is no length limit for python identifier
6. If identifier starts with \_ symbol then it is Private
7. If identifier starts with \_\_ symbol then it is Strongly Private
8. If identifier starts with \_\_ symbol and ends with \_\_ then it is language specific identifier

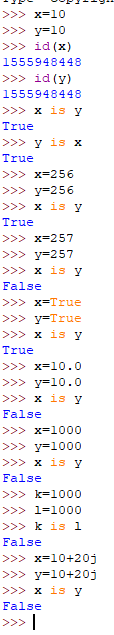
* **Reserved words**
* Words to represent some functionality
* 33 keywords
* True, False, None [3]
* and, or, not, is [4]
* if, else, elif [3] (no switch statement)
* while, for, break, continue, return, in, yield [7]
* try, except [catch block], finally, raise, assert [5]
* import, from, as, class, def, pass, global, nonlocal, lambda[anonymous functions], del, with [11]
* **Data Types**

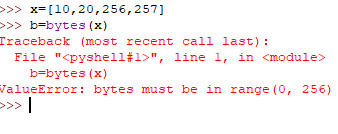
1. Int [*decimal, binary, octal and hexadecimal*]
2. Float [*exponential form is allowed*]
3. Complex [*a.real, b.imag*]
4. Bool [*True, False*]
5. Str [*multi line strings,’’’* *; s[start:end:step], slice operator*] [*+ index & - index* ] *[\*] [len(s)]*
6. Bytes
7. Bytearray
8. Range
9. List
10. Tuple
11. Set
12. Frozenset
13. Dict
14. None

***Everything in python is an object***

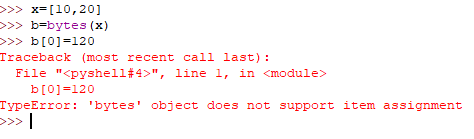
* **Base Conversions**
* Bin()
* Oct()
* Hex()
* **Type Casting or Type Coersion**
* Int()
* *Int(123.56) ==> 123*
* *Int(10+20j) ==> can’t convert*
* *Int(True) ==>> 1*
* *Int(“10”) ==> 10*
* *Int(“10.5”) ==> Type Error, base 10 only allowed*
* Float()
* Complex()
* *Complex(x) ==>> x+0j*
* *Complex(x,y) ==>> x+yj*
* *Complex(True) ==> 1+0j*
* *Complex(“10.5”, “20”) ==>> 10.5 + 20j*
* Str()
* *Str(10) ==>> ‘10’*
* *Str(10+20j) ==> ‘10+20j’*
* Bool()
* *Bool(0) ==>> False*
* *Bool(1) ==>> True*
* *Bool(10) ==>> True ; any non-zero number is True*
* *Bool(0.0)==>> False*
* *Bool(0.1)==>> True*
* *Bool(0+0j) ==>> False*
* *Bool(0+1j) ==>> True ; at least real or imaginary is non-zero then True*
* *Bool(“”) ==> False; empty string case*
* *Bool(“A”)==>> True; any other string even space as string*
* **Immutable Vs Fundamental Data Types**
* Everything in python is object
* All fundamental data types are immutable
* Object creation is not costlier than checking for value in memory
* Memory utilization
* Performance
* **Reusing same object is defined in the following ranges**
* Int ---- 0 to 256
* Bool---always
* Str-----always
* Float---never
* Complex ---never

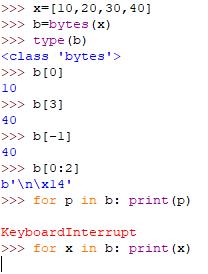
**At the time python interpreter starts or PVM starts:**

* From 0 to 256 objects will be created at the beginning
* Otherwise python start up becomes very late
* String literal at the time of execution or run time only
* For float, the range is infinite between 0 and 1 only [most commonly used]
* 
* **Bytes**
* It represents group of byte numbers just like array
* Values in bytes data type the range of values is 0 to 256

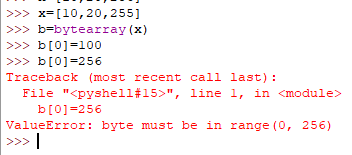


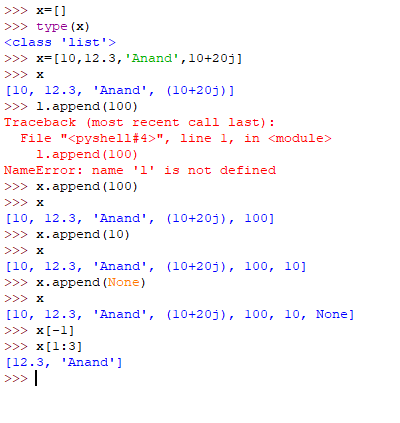
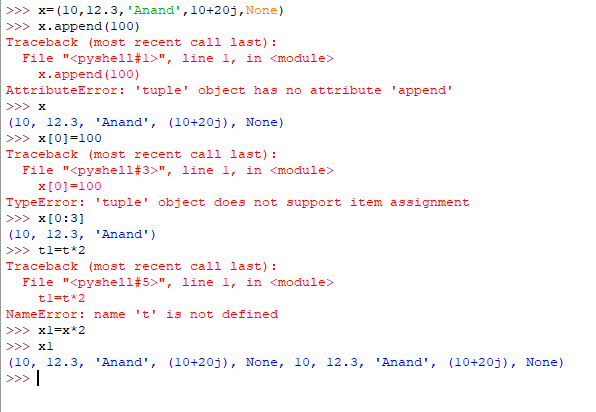
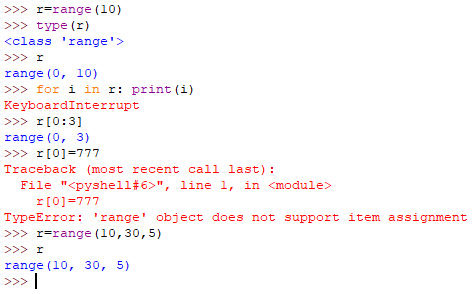
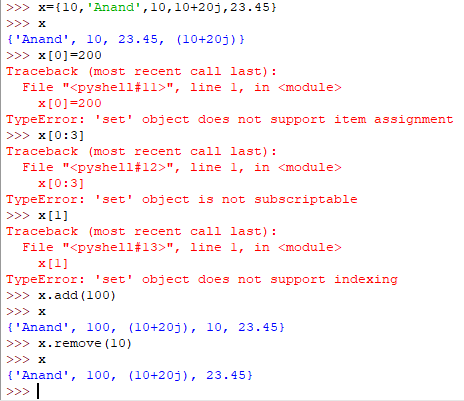
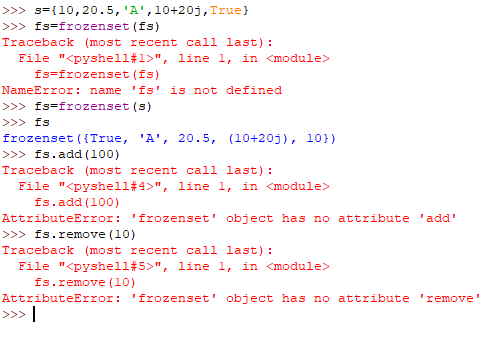
* Bytes objects are immutable and does not support item assignment

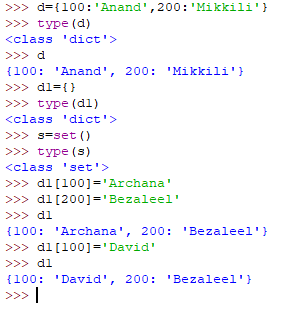


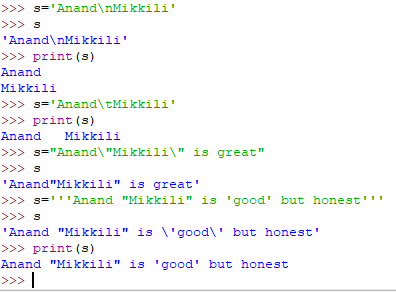


* **Bytearray**
* It represents group of byte numbers just like array
* Values in bytes data type the range of values is 0 to 256
* Bytearray is mutable and supports item assignment

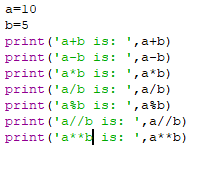
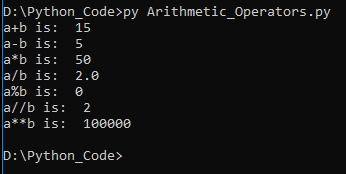
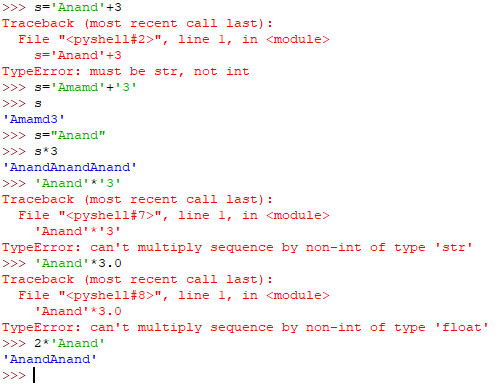


* **List**
* Group of values where insertion order is preserved and duplicates are allowed
* Repeat operator is also allowed [**S\*2**]
* Not immutable
* 
* **Tuple**
* Tuple is same as List in every feature but it is immutable
* 
* **Range**
* Represents a sequence of values
* Elements present inside a Range is always immutable
* It represents values from 0 to end-1 [range(end)]
* Range(from,end-1) [range(10,30) …10 to 29]
* Range(from,end,step) [range(10,30,5) …10,15,20,25]
* Float objects are not allowed since it accepts only integer
* 
* **Set**
* No duplicates and no order of insertion of elements
* Curly braces for set, parenthesis for Tuple and brackets for List
* Index slicing is not applicable for set since no preserved order of elements
* Set is immutable
* 
* **Frozenset**
* Immutable and same like set
* 
* **Dict**
* Key-value pairs
* Key can be any type of object and Value can be any type of object
* Empty with curly braces will be dictionary only; for empty set **s=set()**

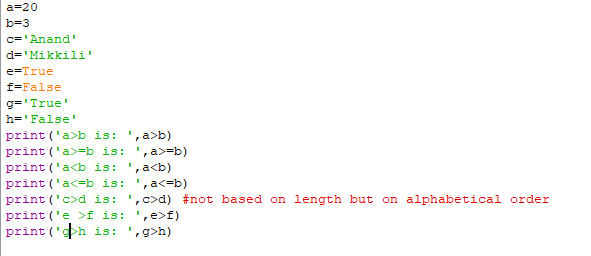
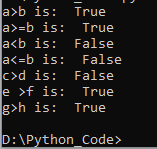


* **None**
* To handle no value, None is introduced
* **Escape Characters**
* **\n,\t[**horizontal tab**], \r [**go to first position**], \b,\f,\’,\”,\\,\v [**vertical tab**]**
* “Anand\”Mikkili” is good”
* ‘Anand “Mikkili” is good’
* “Anand ‘Mikkili’ is good”
* ‘’’Anand ‘Mikkili’ is “good” but honest’’’
* “””Anand ‘Mikkili’ is “good” but honest”””
* ****
* **Constants**
* There is no way that we can assign a value as constant. So constant concept but developer only has to take care of it. At language level there is no concept
* **Operators**

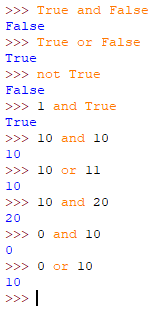
1. Arithmetic Operators

* +, -, \*, /, %, // [floor division], \*\* [exponential or power]
* 
* 
* + Operator for String then both the arguments must be string
* \* Operator for String then String multiplication
* 
* x/0 or x%0 or x//0; Zero division error

1. Relational Operators or Comparison operators

* >, >=, <, <=, ==, !=
* 
* 
* Chaining of relational operators is possible
* If all the comparisons are True then final answer is True [**10<20<30 ….True**]
* If one of the comparison is False; then the result is False for whole chain [**10>20<30<40<50…False**]
* == is content comparison whereas is keyword is address comparison
* Chaining of relation is applicable for ==
* A

1. Logical Operators

* and, or, not
* Apply on Boolean type result is True or False
* Non-Boolean Type:
* [x and y]; If x evaluates to False then result is x otherwise return y
* [x and y]; If x evaluates to False then return y otherwise return x
* 

1. Bitwise Operators

* &, |, ^[**xor**], ~[**bit wise compliment**], << [left shift], >> [right shift]
* Bitwise operators are only applicable for boolean and integer data types
* If both the bits are 1 then 1 otherwise 0 [&]
* If at least one bit is 1 then 1 otherwise 0 [ | ]
* If both arguments are same then 0 otherwise 1 [ ^ ]
* ~
* Positive numbers will be represented normally
* Negative numbers represent in 2’s complement form
* Replace 0’s with 1 and 1’s with 0 to achieve 1’s complement form
* Add 1 to least significant bit to achieve 2’s complement form

~4

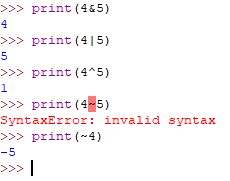
0000..100

1111...011 --negative number so convert into 2’s complement form

0000..100

1

0000...101 -🡪 -5

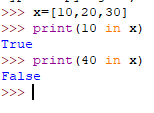
* A
* 1 ==>0 and 0 ==> 1
* Bitwise left shift [<<]
* Bitwise right shift [ >>]
* 

1. Assignment Operators

* x, y=10,20.5
* compound assignment operator x **+=**10
* ++x and x++ are not available in python [increment and decrement]

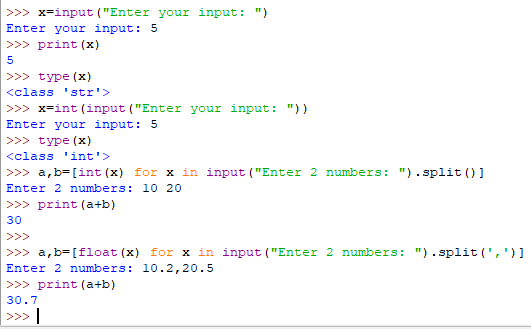
1. Special Operators

* Identity Operators
* ***Is*** [a is b; if a and b are pointing to same object then True]
* ***Is not***
* Membership Operators
* Whether the object is member of specified user defined data type or not
* ***In*** and ***not in*** are keywords for membership check

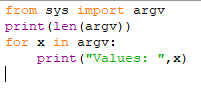


1. Operator precedence

* Highest priority goes to parenthesis ( ***( )*** )
* ***\*\**** operator will get next highest priority
* Unary operator will get next highest priority [ **Ex: ~x, -x unary minus**]
* Binary operators will get next highest priority [**\*, /, %, //, +, -** ; **x+y** ]
* Ternary operators will get next highest priority [**<<, >>, &, ^, |, >, >=, <, <=, ==, !=, =, +=, -=, \*=, is, is not, in, not in, not, and, or**]
* Assignment will get least priority
* **Module**
* Module is a low level component which contains a group of functions, variables and classes [***ex: math module; like in java math class***]
* Library is high level component which contains modules
* Aliasing is available in python for modules are libraries [***import math as m***]; Once aliasing is done you should not refer the original name
* Import directly the function instead of module in python [***from math import sqrt; from math import \****]
* **Input & Output Statements**

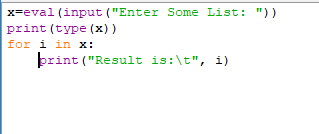


* **Command Line Arguments**



* **Eval()**

Command line arguments without type casting



* **Flow Control**

1. Conditional Statements/Selection Statements
2. If
3. If-else
4. If-elif-else

Note:

* **Else is always optional**

1. Iterative Statements
2. For [for-else is also possible]

* For-else: only when loop executes without break and after loop completes then else block will be executed

1. While [while-else is also possible]
2. Transfer Statements
3. Break

* Break the loop and come out

1. Continue

* Skip current iteration and go for next iteration
* We’ll use continue only inside loops

1. Pass

* Wherever an empty block is required in if condition or elif condition or else then we use pass statement which do not perform anything
* Empty statement or Null statement or nothing. This is just to support syntactic existence
* This can be used in **conditional statements** and **functions**

1. Del keyword

x=10

del x

* del vs None
* del: I do not need both object reference and also object
* None: I do not need object but might need reference for future use