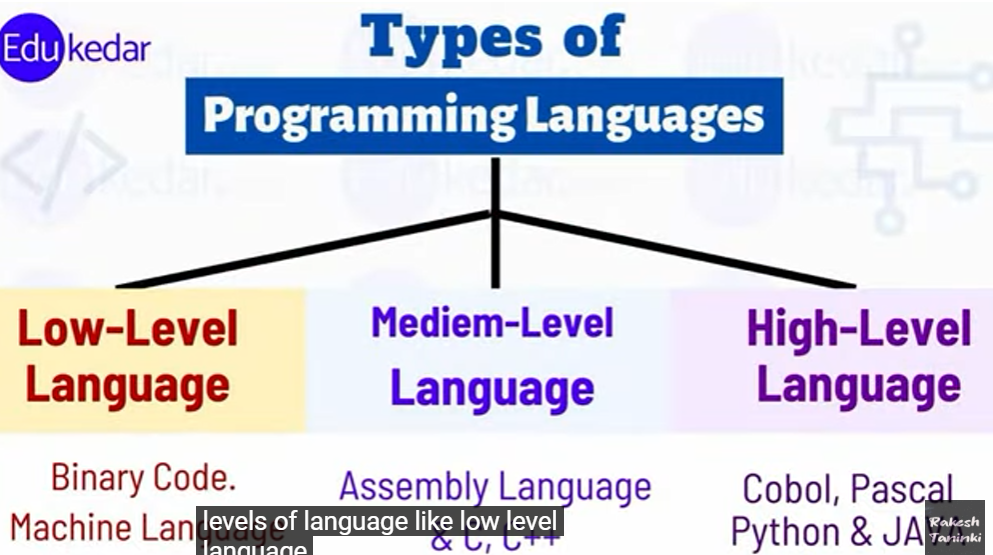
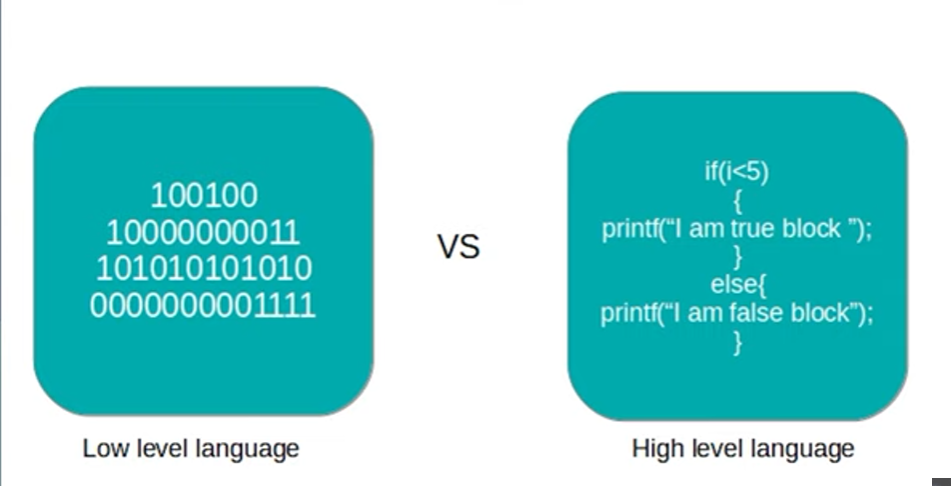
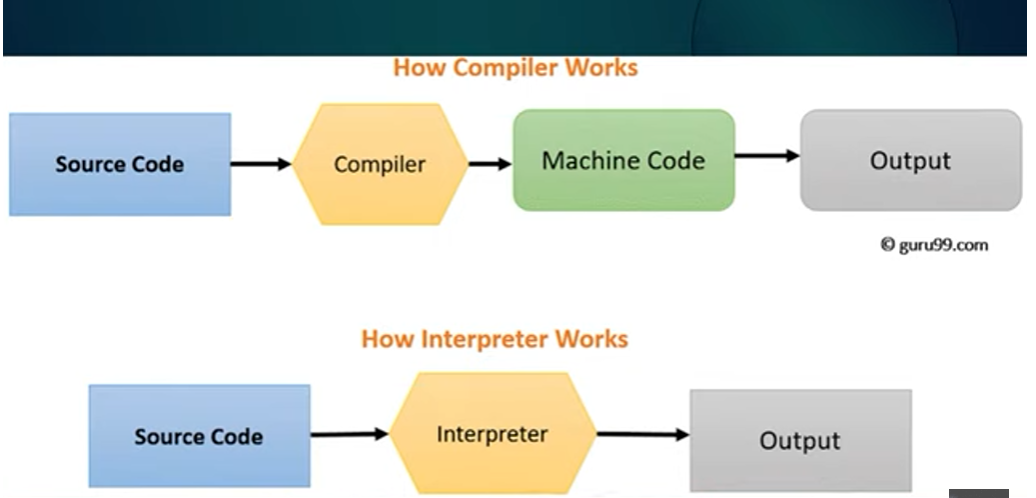
Guido vann Rosum author of python in 1989 it was devoloped







1. Python is a high level language
2. Interpreted language
3. In interpreter no need of compiler to covert source code to machine understandable code directly source code gets executed through interpreter and gives the output
4. Object oriented programming language
5. Python from starting its object oriented language
6. Python version 2.7 ,3.6,3.7,3.8,3.9,3.10,3.11 2.7 is not getting supported
7. 3.12 is going to release in oct

Print() this is a function

Esc function \ to ignore that words

\n to print next line

\n\t to print next lin with space

**Comments:**

1)Comments starts with a #, and Python will ignore them:

Ex: #This is a comment  
print("Hello, World!")

2) Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

**Example:**

"""  
This is a comment  
written in  
more than just one line  
"""  
print("Hello, World!")

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

x = 5  
y = "John"  
print(x)  
print(y)

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

Double quotes is same as single quotes

X=”john”

Y=’john’

Print(x)

Print(y)

Both has same results

### **Example**

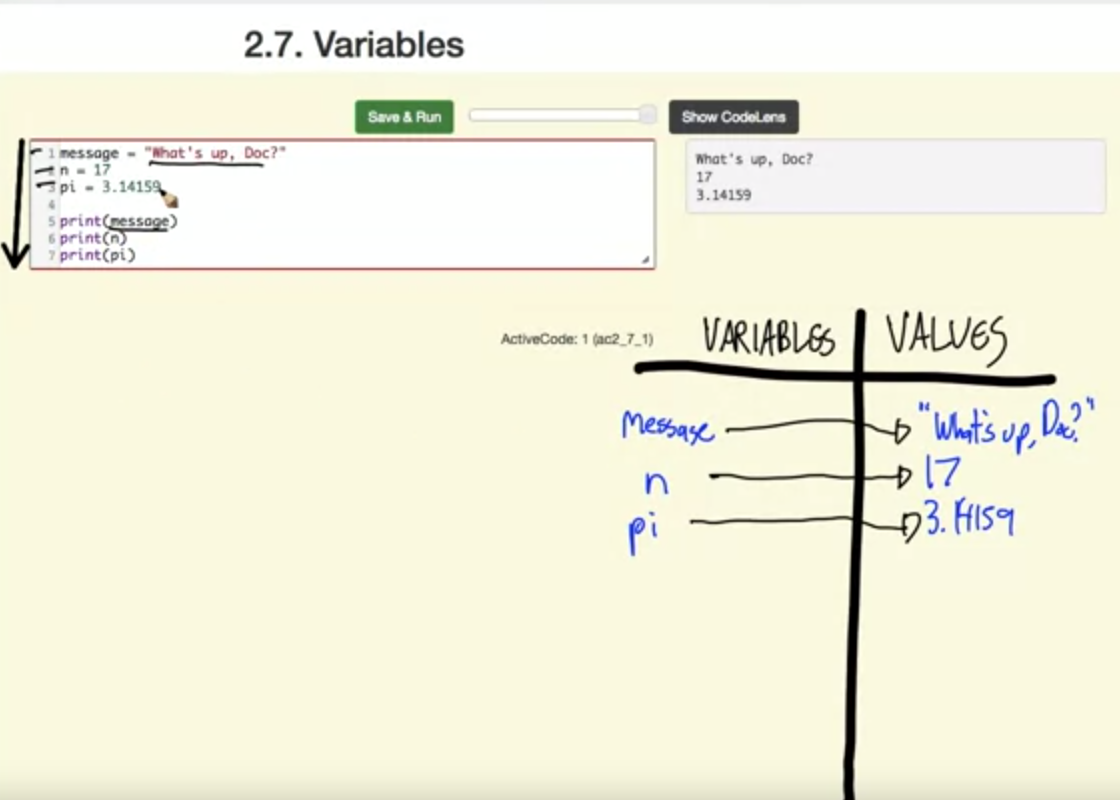
This will create two variables:

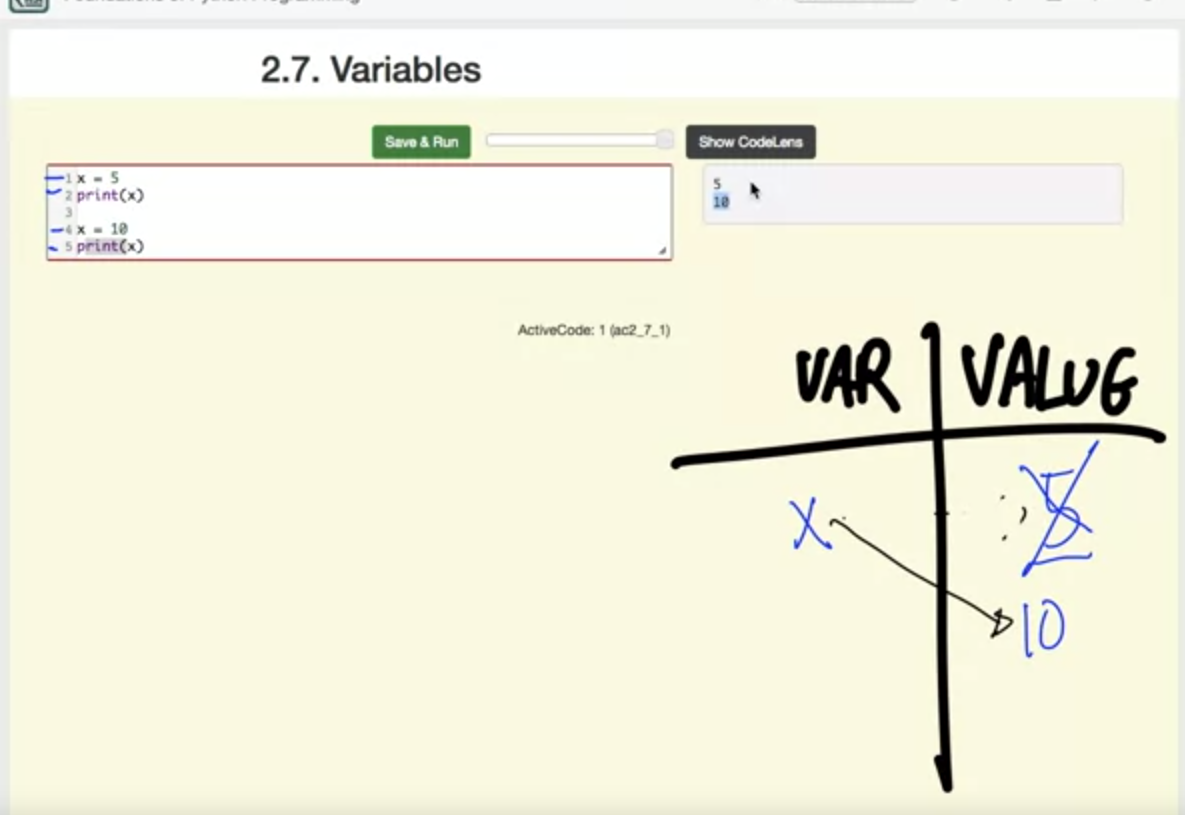
a = 4  
A = "Sally"  
#A will not overwrite a

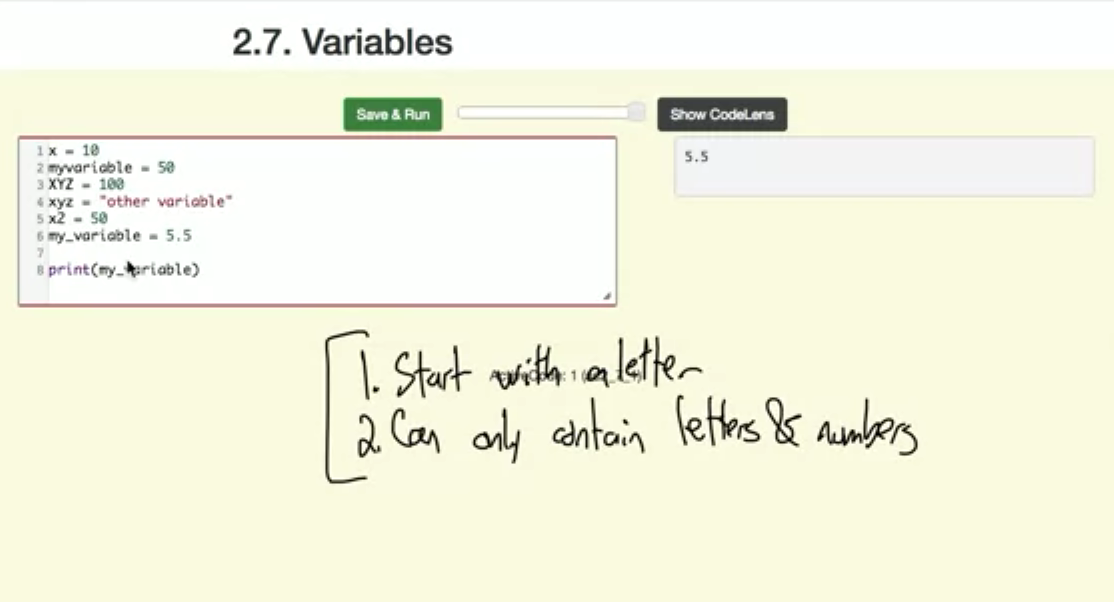
## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

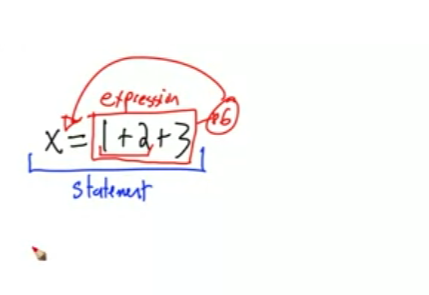




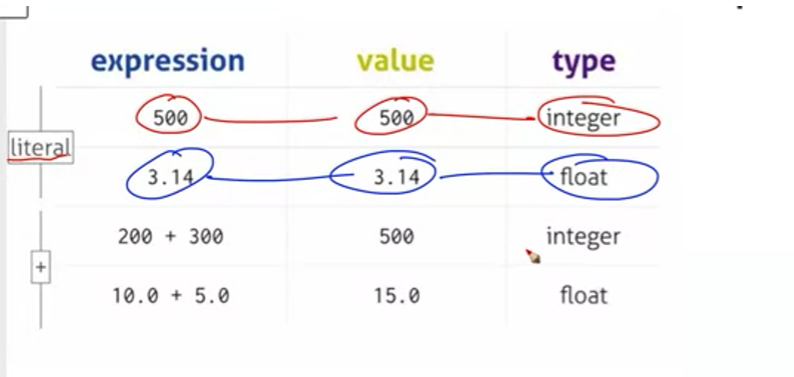


**Python keywords :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| and | as | assert | break | class | continue |
| def | del | elif | else | except | exec |
| finally | for | from | global | if | import |
| in | is | lambda | nonlocal | not | or |
| pass | raise | return | try | while | With |
| yield | True | False | None |  |  |



**6 is value ,x is variable**



**Example :**

myvar = "John"  
my\_var = "John"  
\_my\_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"

## Multi Words Variable Names

Variable names with more than one word can be difficult to read.

There are several techniques you can use to make them more readable:

## Camel Case

Each word, except the first, starts with a capital letter:

myVariableName = "John"

## Pascal Case

Each word starts with a capital letter:

MyVariableName = "John"

## Snake Case

Each word is separated by an underscore character:

my\_variable\_name = "John"

## Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:

### **Example**[**Get your own Python Server**](https://www.w3schools.com/spaces/)

x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables8)

**Note:** Make sure the number of variables matches the number of values, or else you will get an error.

## One Value to Multiple Variables

And you can assign the same value to multiple variables in one line:

### **Example**

x = y = z = "Orange"  
print(x)  
print(y)  
print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables6)

## Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

### **Example**

Unpack a list:

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits  
print(x)  
print(y)  
print(z)

myname="sravya"

print(myname)

myFirstName="sravya2" #Camel case

MyFirstName="Sravya3" #Pascal case

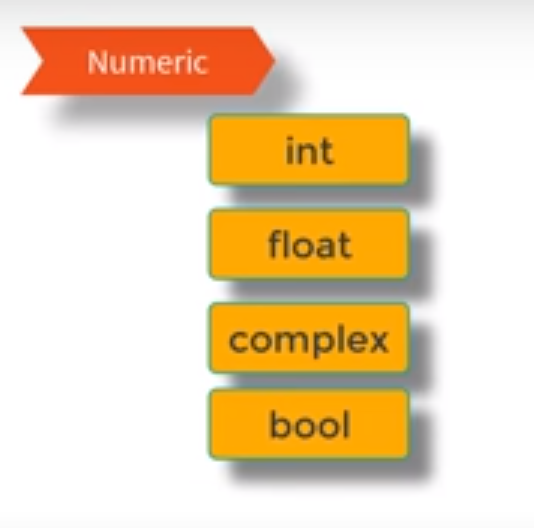
my\_first\_name="Sravya 3" # snakecase

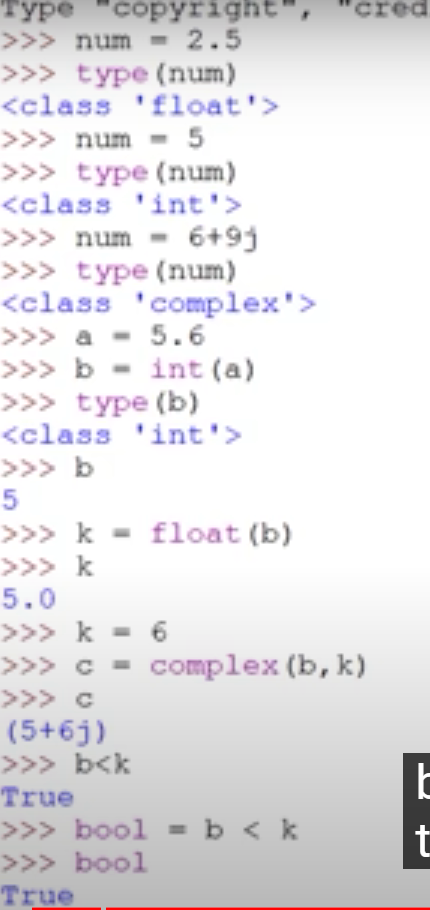
Data types in python



**None** is a null value assigned to a variable

**Numeric is again classified into following types**





**Complex numbers are written with a "j" as the imaginary part**:

### **Example**

Complex:

x = 3+5j  
y = 5j  
z = -5j  
  
print(type(x))  
print(type(y))  
print(type(z))

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**Set**](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.
* List which is ordered and chageable it allows duplicate values as well and it supports multiple datatypes
* **Lists** are mutable that means you can change the values
* **Tuples** supports different datatype its similar to lists but it cannot be changeable i.e immutable
* List =[] supports append,pop,remove, del,index,count,insert
* Tup=() supports only count and index
* We use tuples in such a case that u want list but don’t want to change the list in that kind of situation we use tuples
* Iteration is little faster in tuples when compare list as they are immutable
* Allows duplicate values

**Set : { }**  is unordered unindexed and doesn’t allows duplicate values

**Dictionary** :{} which is similar to sets but it follows key and values pair to fetch the data without indexing

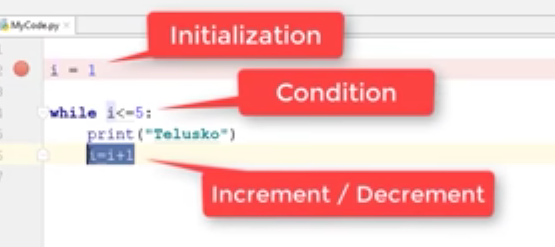
You have to specify key to fetch the values

Python has a set of built-in methods that you can use on sets.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two or more sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with another set, or any other iterable |

Loops: in python

1)while loop



2)for loop

There is no initialization ,condition,incremental to be specified its automatical done by for

X=[2.0,5,8,’sravya’]

For i in x to print 1 by one value in x it represents one element of list at a time

For loop is used for sequence we can use it with strings,tuples, list

Python operators

+ ,-,/,%,\*,(\*\*)-> square

4 \*\* 2 ==16 4square

**Order of operators**

1)( )

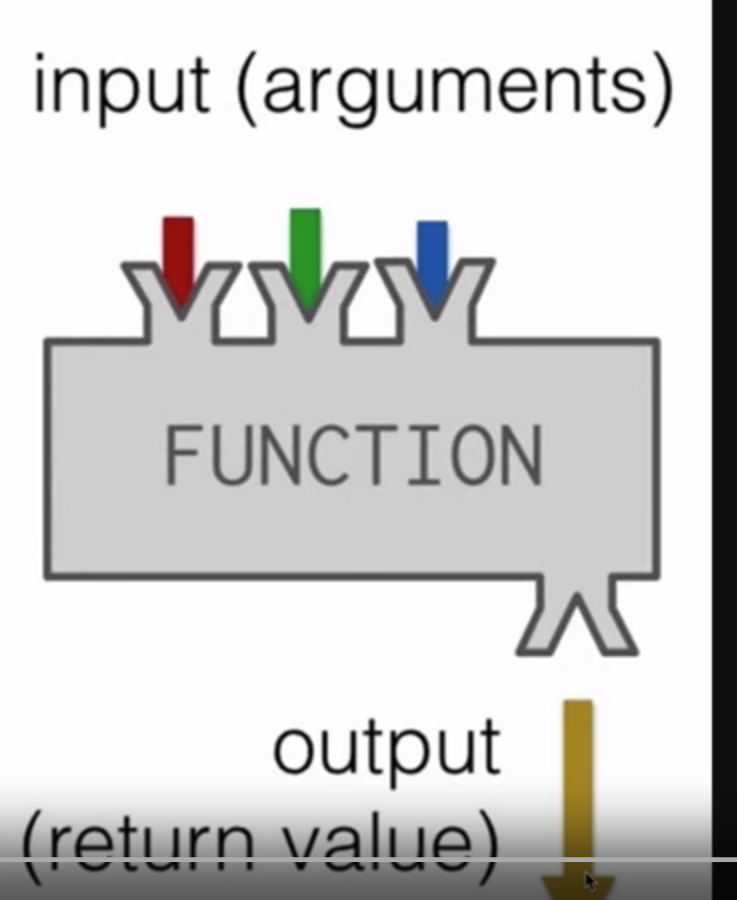
2)\*\* square exponents

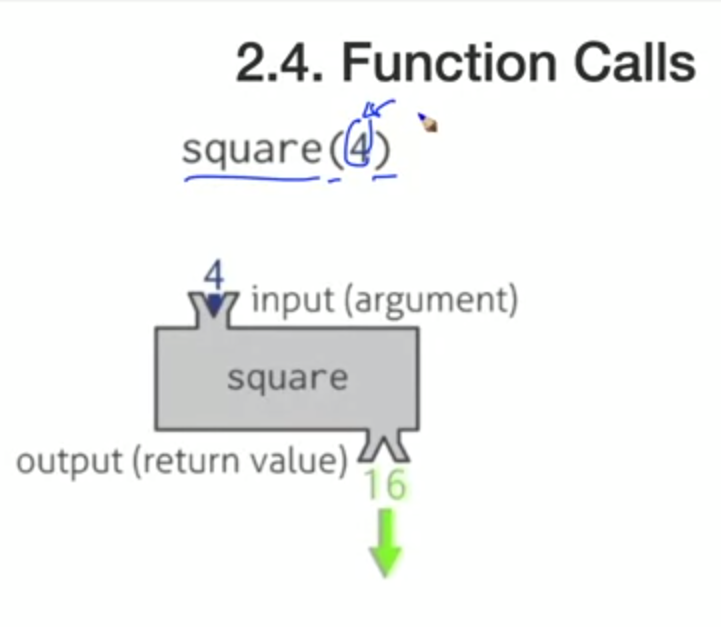
3) \* /

4) + -

1. Parentheses have the highest precedence and can be used to force an expression to evaluate in the order you want. Since expressions in parentheses are evaluated first, 2 \* (3-1) is 4, and (1+1)\*\*(5-2) is 8. You can also use parentheses to make an expression easier to read, as in (minute \* 100) / 60: in this case, the parentheses don’t change the result, but they reinforce that the expression in parentheses will be evaluated first.
2. Exponentiation has the next highest precedence, so 2\*\*1+1 is 3 and not 4, and 3\*1\*\*3 is 3 and not 27. Can you explain why?
3. Multiplication and both division operators have the same precedence, which is higher than addition and subtraction, which also have the same precedence. So 2\*3-1 yields 5 rather than 4, and 5-2\*2 is 1, not 6.
4. Operators with the same precedence are evaluated from left-to-right. In algebra we say they are left-associative. So in the expression 6-3+2, the subtraction happens first, yielding 3. We then add 2 to get the result 5. If the operations had been evaluated from right to left, the result would have been 6-(3+2), which is 1.

**Functions**





If we call a function and gives the arguments then only it gets executed

* In functions how many arguments its going to take it returns only 1 value as a output