C# S
COD
EXER
CODING
WITH H
WRITTEN
JJ TAM
EDITED BY
TAM SEL

JAVA
CODING
EXERCI
CODING FOR
WITH HANDS
WRITTEN BY
JJ TAM
EDITED BY
TAM SEL

LEARN PYTHON QUICKLY

CODING FOR BEGINNERS WITH HANDS ON PROJECTS

python

WRITTEN BY
JJ TAM

EDITED BY
TAM SEL

LEARN PYTHON QUICKLY AND CODING EXERCISES PYTHON, JAVA AND C#

CODING FOR BEGINNERS
WITH HANDS ON PROJECTS
BY
J J TAM

LEARN PYTHON QUICKLY

CODING FOR BEGINNERS
WITH HANDS ON PROJECTS

BY
J J TAM

LEARN PYTHON QUICKLY

Well it is almost 5 years I started my IT career, I love to work in Java, and explore open source packages. Now a day I felt bored to work in Java, so want to learn new language, one of my friend suggested me to learn python. Immediately I thought, why should I learn python?

Following are some of the reasons that forced me to learn python… (First and best reason is I am fed up in working with same language for 5 years……….:))

a. Python is open source and is available on Windows, Mac OS X, and Unix operating systems. You can download and work with python for free.

b. Python is fun to experiment and easy to learn. Python provides interpreter that can be used interactively, which makes it easy to experiment with features of the language.

c. You can extend the python interpreter with new functions and data types implemented in C or C++

d. Python improves developer productivity many times as compared to C, C++ and Java.

e. No need to compile python program, it runs immediately after development.

f. We can easily port python project from one platform to another (Of course Java also provides this feature)

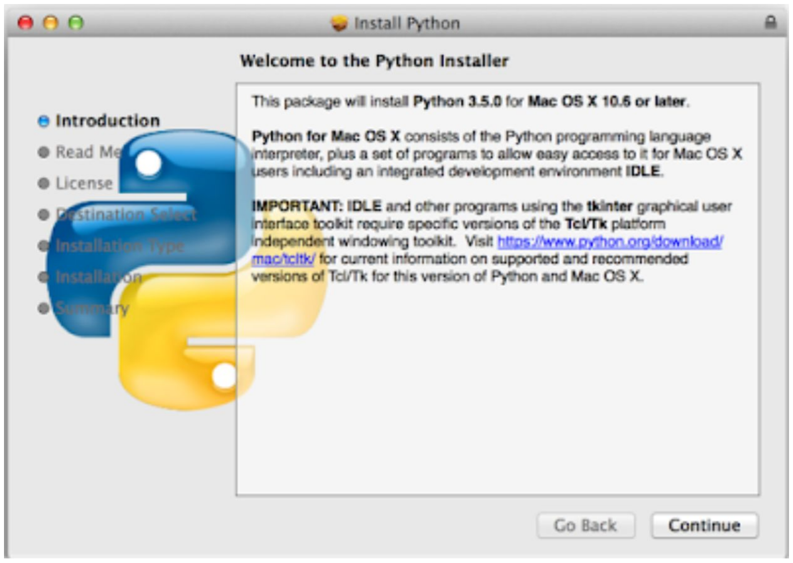
g. Rich in built library. Many third party open source libraries are available for logging, web development (Django — the popular open source web application framework written in Python), networking, database access etc.

h. Python has very great community, whatever the problem you faced in python, you will get quick help.

Install python on MAC OS

Step 1: Download python software from following location. I downloaded pkg file to install on mac os.
https://www.python.org/downloads/

Step 2: Click the pkg file.



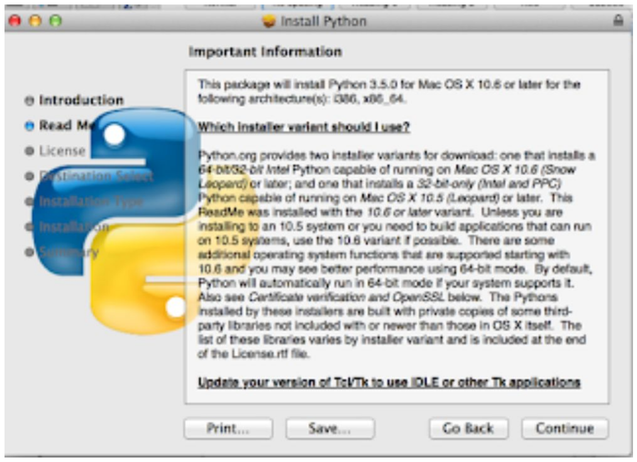
Welcome to the Python Installer

This package will install **Python 3.5.0** for **Mac OS X 10.6 or later**.

Python for Mac OS X consists of the Python programming language interpreter, plus a set of programs to allow easy access to it for Mac OS X users including an integrated development environment **IDLE**.

IMPORTANT: IDLE and other programs using the **tkinter** graphical user interface toolkit require specific versions of the **Tcl/Tk** platform independent windowing toolkit. Visit https://www.python.org/download/mac/tcltk/ for current information on supported and recommended versions of Tcl/Tk for this version of Python and Mac OS X.

Press Continue.



Important Information

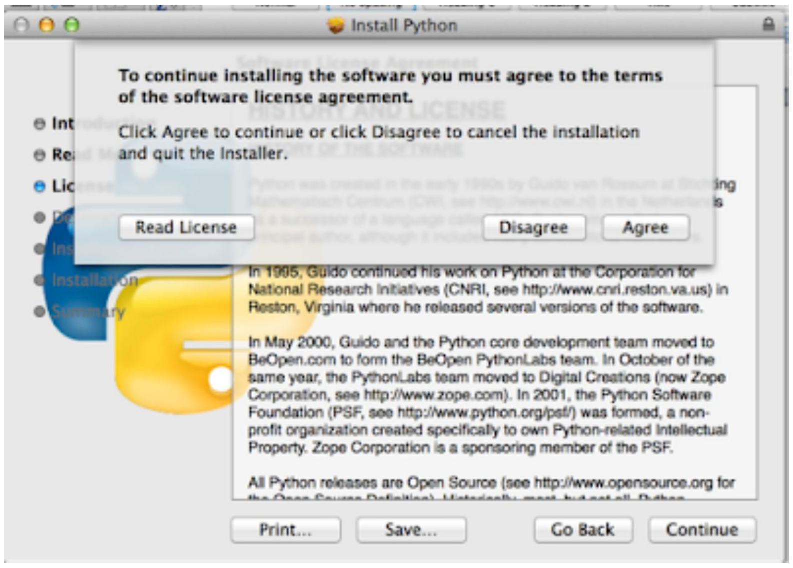
This package will install Python 3.5.0 for Mac OS X 10.6 or later for the following architecture(s): i386, x86_64.

Which installer variant should I use?

Python.org provides two installer variants for download: one that installs a *64-bit/32-bit Intel* Python capable of running on *Mac OS X 10.6 (Snow Leopard)* or later; and one that installs a *32-bit-only (Intel and PPC)* Python capable of running on *Mac OS X 10.5 (Leopard)* or later. This ReadMe was installed with the *10.6 or later* variant. Unless you are installing to an 10.5 system or you need to build applications that can run on 10.5 systems, use the 10.6 variant if possible. There are some additional operating system functions that are supported starting with 10.6 and you may see better performance using 64-bit mode. By default, Python will automatically run in 64-bit mode if your system supports it. Also see *Certificate verification and OpenSSL* below. The Pythons installed by these installers are built with private copies of some third-party libraries not included with or newer than those in OS X itself. The list of these libraries varies by installer variant and is included at the end of the License.rtf file.

Update your version of Tcl/Tk to use IDLE or other Tk applications

Press Continue.



To continue installing the software you must agree to the terms of the software license agreement.

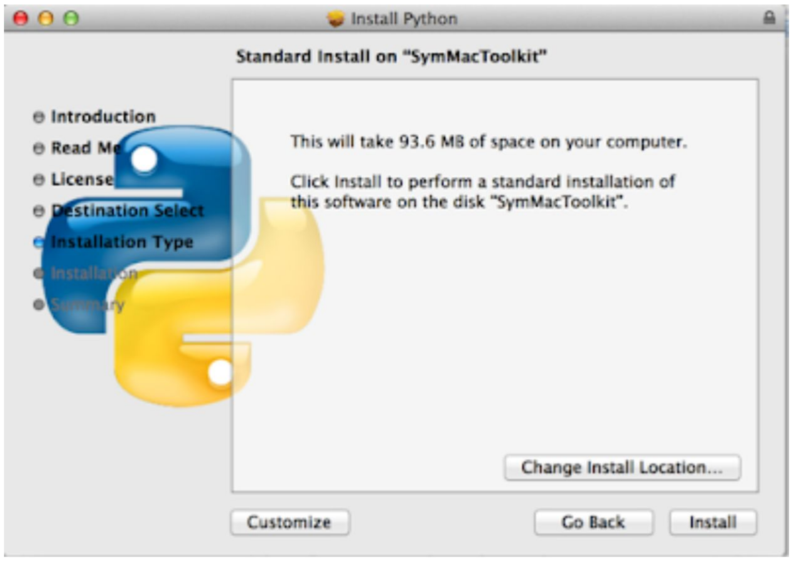
Click Agree to continue or click Disagree to cancel the installation and quit the Installer.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for

Accept license agreement and press Continue.

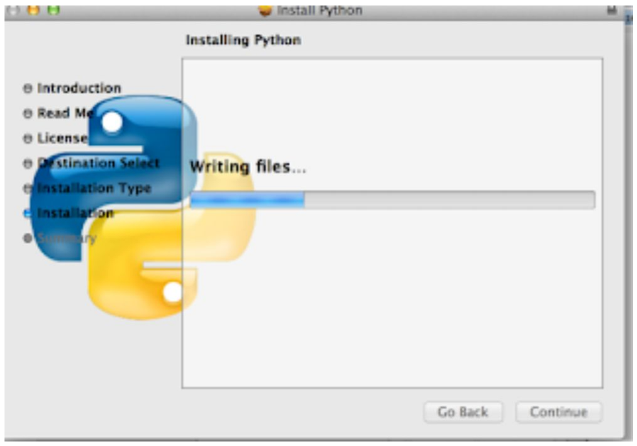


Standard Install on "SymMacToolkit"

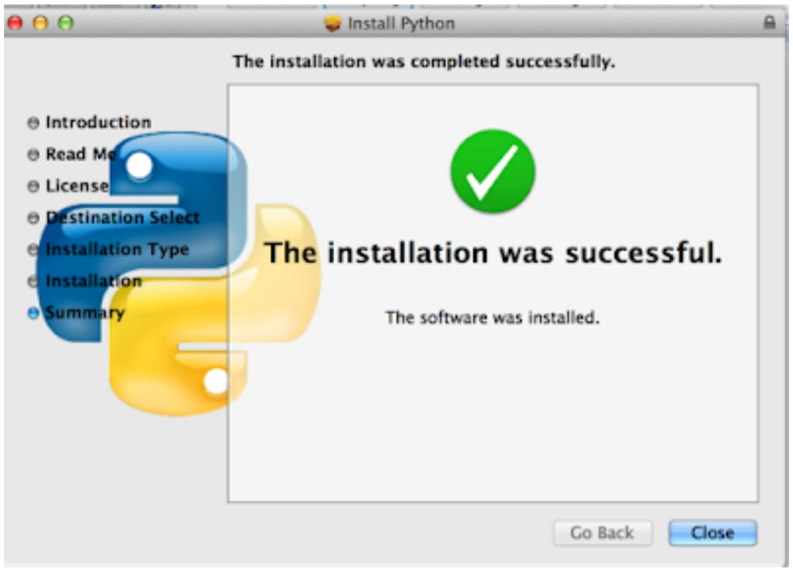
This will take 93.6 MB of space on your computer.

Click Install to perform a standard installation of this software on the disk "SymMacToolkit".

You can change the installation location use the button 'Change Install Location', and press the button Install.



Once installation is successful, you will get following screen.



Once installation is successful, open terminal and type python3 (Since I installed python 3.5).

```
$ python3
Python 3.5.0 (v3.5.0:374f501f4567, Sep 12 2015, 11:00:19)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>  quit()
```

Use ‘quit()’ to exit from python prompt.

Location of python3

```
$ which python3
/Library/Frameworks/Python.framework/Versions/3.5/bin/python3

$ which python3.5
/Library/Frameworks/Python.framework/Versions/3.5/bin/python3.5
```

On windows machines python usually placed at ‘C:\Python35’.

Python: Hello World program

Open any text editor and copy the statement “print ('hello world')” and save the file name as hello.py.

hello.py

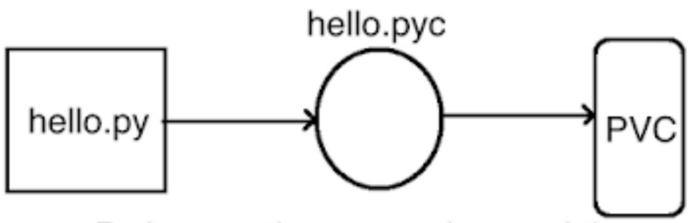
```
print ('hello world')
```

Open terminal (or) command prompt, use the command ‘python3 hello.py’ to run the file hello.py. You will get output like below

```
$ python3 hello.py
hello world
```

What happens when you instruct python to run your script?

Python first compiles your source code to byte code and sends it to python virtual machine (PVM). Byte code is the platform independent representation of your source code. PVM reads the byte code one by one and execute them.



Python runtime execution model

Note:

Byte code is not machine understandable code, it is python specific representation.

Python interactive command line

Open command prompt (or) terminal and type ‘python3’ command. It opens python interactive session.

```
$ python3
Python 3.5.0 (v3.5.0:374f501f4567, Sep 12 2015, 11:00:19)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

How to exit from python interpreter

On Windows, a Ctrl-Z gets you out of this session; on Unix, try Ctrl-D instead. Another way is simply call the quit() function to quit from python.

```
$ python3
Python 3.5.0 (v3.5.0:374f501f4567, Sep 12 2015, 11:00:19)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
$
```

You can type any statement (or) expressions in python interpreter, interpreter execute those statements and gives you the result.

```
>>> 10+20
30
>>> 20*20
400
```

```
>>> print('Hello World')
Hello World
```

Note:

On Unix, the Python 3.x interpreter is not installed by default with name python, so that it does not conflict with other installed Python 2.x executable.

Python: Operators

An operator is a symbol, which perform an operation. Following are the operators that python supports.

Arithmetic Operators

Relational Operators

Assignment Operators

Logical Operators

Bitwise Operators

Membership Operators

Identity Operators

Arithmetic Operators in python

Following are the arithmetic operators supported by python.

Operator	Description	Example
+	Addition	2+3
-	Subtraction	2-3
*	Multiplication	2*3
/	Division always returns a float value. To get only integer result use //.	2/3 returns 0.6666666666666666
//	Floor division discards the fractional part	2//3 returns 0
%	Returns the remainder of the division.	2%3
**	Power	2**3 returns 2 power 3 = 8

```
>>> 4+2
6
>>> 4-2
```

```
2
>>> 4/2
2.0
>>> 4*2
8
>>> 4//2
2
>>> 4**2
16
```

Relational Operators in python

Relational operators determine if one operand is greater than, less than, equal to, or not equal to another operand.

Operator	Description	Example
==	Equal to	a==b returns true if a is equal to b, else false
!=	Not equal to	a!=b returns true if a is not equal to b, else false.
>	Greater than	a>b returns true, if a is > b, else false.
>=	Greater than or equal to	a>=b returns true, if a is >= b, else false.
<	Less than	a<b returns true, if a is < b, else false.
<=	Less than or equal to	a<=b returns true, if a is <= b, else false.

```
>>>  a =10
>>>  b =12
>>>  a == b
False
```

```
>>>  a != b
True
>>>  a > b
False
>>>  a >= b
False
>>>  a < b
True
>>>  a <= b
True
>>>
```

Assignment operators in python

Assignment operators are used to assign value to a variable. Following are the assignment operators provided by python.

Operator	Description
=	a=10 assigns 10 to variable a
+=	a+=10 is same as a=a+10
-=	a-=10 is same as a=a-10
*=	a*=10 is same as a=a*10
/=	a/=10 is same as a=a/10
//=	a//=10 is same as a=a//10
%=	a%=10 is same as a=a%10
**=	a**=10 is same as a=a**10

```
>>> a =10
>>> a
10
>>>
>>> a +=10
>>> a
20
>>>
>>> a -=10
>>> a
10
>>>
>>> a *=10
>>> a
```

```
100
>>>
>>> a /=10
>>> a
10.0
>>>
>>> a //=10
>>> a
1.0
>>>
>>> a **=10
>>> a
1.0
```

Multiple Assignments

You can assign values to multiple variables simultaneously.

```
>>> a, b, c  =  10 ,  'hello' ,  12.345
>>> a
10
>>> b
'hello'
>>> c
12.345
```

Logical Operators in python

Following are the logical operators supported by python.

Operator	Description
and	‘a and b’ returns true if both a, b are true. Else false.
or	‘a or b’ return false, if both a and b are false, else true.
not	‘not a’ Returns True if a is false, true otherwise

```
>>> a = bool( 0 )
>>> b = bool( 1 )
>>>
>>> a
False
>>> b
True
>>>
>>> a and b
False
>>>
>>> a or b
True
>>>
>>> not a
True
>>>
>>> not (a and b)
```

```
True
>>>
>>> not (a or b)
False
```

Bitwise Operators in python

Python supports following bitwise operators, to perform bit wise operations on integers.

Operator	Description
>>	bitwise right shift
<<	bitwise left shift
&	bitwise and
^	Bitwise Ex-OR
|	Bitwise or
~	Bitwise not

Following post, explains bitwise operators clearly.
http://self-learning-java-tutorial.blogspot.in/2014/02/bit-wise-operators.html

```
>>> a =2
>>> a >>1
1
>>>
>>> a <<1
4
>>>
>>> a &3
2
>>> a |3
3
>>>
```

```
>>> ~ a
-3
```

Membership operators in python

Membership operators are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

Operator	Description
in	Return true if value/variable is found in the sequence, else false.
not in	Return True if value/variable is not found in the sequence, else false

```
>>> primeNumbers = [ 2 , 3 , 5 , 7 , 11 ]
>>> 2 in primeNumbers
True
>>> 4 in primeNumbers
False
>>> 4 not in primeNumbers
True
```

Identity operators in python

Identity operators are used to compare the memory locations of two objects.

Operator	Description
is	a is b returns true, if both a and b point to the same object, else false.
is not	a is not b returns true, if both a and b not point to the same object, else false.

```
>>> name1 ="Hari Krishna"
>>> name2 = name1
>>> name3 ="Hari Krishna"
>>> name4 ="abc"
>>>
>>> name1  is  name2
True
>>> name1  is  name3
False
>>> name1  is  name4
False
>>> name1  is  not  name3
True
```

Python: Short circuit operators

Boolean operators and, or are called short circuit operators, it Is because evaluation of expression stops, whenever the outcome determined.

Why Boolean and is called short circuit operator?
Since if the first statement in the expression evaluates to false, then python won't evaluates the entire expression. So boolean and is called short circuit and.

Why Boolean OR is called short circuit operator?
Since if the first statement evaluates to true, then Python won't evaluates the entire expression.

Strings in python

String is a sequence of character specified in single quotes('…'), double quotes("…"), (or) triple quotes ("""…""" or '''… '''). Strings in python are immutable, i.e., an object with a fixed value.

```
>>> str1 ='Hello World'
>>> str2 ="Hello World"
>>> str3 ="""Hello
... World
... """
>>> str1
'Hello World'
>>> str2
'Hello World'
>>> str3
'Hello\nWorld\n'
```

Special characters are escaped with backslash.

```
>>> message ='He don\'t know about this'
>>> message
"He don't know about this"
```

'print' method treat characters preceded by \ (backslash) as special characters.

```
>>> print ( 'firstline\nsecondline' )
firstline
secondline
```

As you observe output, \n prints new line. If you don't want characters prefaced by \ to be interpreted as special characters, you can use raw strings by adding an r before the first quote

```
>>> print ( r'firstline\nsecondline' )
```

```
firstline\nsecondline
```

Concatenate strings
'+' operator is used to concatenate strings.

```
>>> hello ="Hello,"
>>> message ="How are you"
>>> info = hello + message
>>> info
'Hello,How are you'
```

Two or more string literals next to each other are automatically concatenated.

```
>>> 'Hello' "How" 'are' 'you'
'HelloHowareyou'
```

Repeat strings
By using ‘*’, we can repeat the strings.

```
>>> 'hi'*2
'hihi'
>>> 'hi'*2+'hello'*3
'hihihellohellohello'
```

Access specific character from strings
You can access, specific character of string using index position.

```
>>> name ="Hello World"
>>> name[ 0 ]
'H'
>>> name[ 6 ]
'W'
```

Index 0 represents 1st character, 1 represents 2nd character etc., You can also use negative numbers for indexing.

-1 represents last character; -2 represents second-last character etc.,

```
>>> name
'Hello World'
>>> name[ -1 ]
'd'
>>> name[ -2 ]
'l'
>>> name[ -7 ]
'o'
```

Slicing
Slicing is used to get sub string of given string.

Example	Description
string[start:end]	Returns sub string from index start (included) to end index (excluded).
string[:end]	Returns sub string from index 0(included) to end index (excluded).
string[start:]	Return sub string from index start to till end.
string[-2:]	Return characters from 2nd last to end.

```
>>> message ="Hello World"
>>>
```

```
>>> message[ 0 : 5 ]
'Hello'
>>> message[ 5 :]
' World'
>>> message[:]
'Hello World'
>>> message[ -2 :]
'ld'
>>> message[ -5 : -2 ]
'Wor'
```

Get length of the string

Function ‘len(str)’ is used to get the length of the string.

```
>>> message
'Hello World'
>>> len(message)
11
>>> len( "How are you" )
11
```

Python: if condition

“if” statement
"if" tell the program execute the section of code when the condition evaluates to true.

Syntax
if_stmt ::= "if" expression ":" suite

test.py
```
a =10

if (a < 10) :
 print("a is less than 10")

if (a == 10) :
 print("a is equal to 10")

if( a > 10) :
 print("a is greater than 10")
```

```
$ python3 test.py
a is equal to 10
```

if-else statement
If the condition true, then if block code executed. other wise else block code executed.

Syntax
```
if_stmt ::=  "if" expression ":" suite
             ["else" ":" suite]
```

test.py
```
a =10

if (a != 10) :
 print("a is not equal to 10")
else :
 print("a is equal to 10")
```

if-elif-else statement
By using if-elif-else construct, you can choose number of alternatives. An if statement can be followed by an optional elif...else statement.

Syntax

```
if_stmt ::=  "if" expression ":" suite
             ( "elif" expression ":" suite )*
             ["else" ":" suite]
```

test.py

```
a =10

if (a > 10) :
 print("a is greater than 10")
elif (a < 10) :
 print("a is less than 10")
else :
 print("a is equal to 10")
```

```
$ python3 test.py
a is equal to 10
```

Note:
a. In python, any non-zero integer value is treated as true; zero is false.

test.py

```
if (100) :
 print("Hello")
```

```
else :
 print("Bye")
```

```
$ python3 test.py
Hello
```

b. Any sequence (string, list etc.,) with a non-zero length is true, empty sequences are false.

test.py

```
list=[]
data="abc"

if (list) :
 print("list is not empty")
else :
 print("list is empty")

if (data) :
 print("data is not empty")
else :
 print("data is empty")
```

```
$ python3 test.py
list is empty
data is not empty
```

Python: while statement

‘while’ statement executes a block of statements until a particular condition is true

Syntax

```
while_stmt ::=  "while" expression ":" suite
                ["else" ":" suite]
```

‘else’ clause is optional. If the expression evaluates to false, then else clause will execute (if else present), and terminates.

test.py

```
a =2

print ( "Even numbers are" )
while(a < 10) :
 print (a, end=' ')
a +=2
else:
 print("\nExit from loop")

print ( "Done" )
```

```
$ python3 test.py
Even numbers are
2 4 6 8
Exit from loop
Done
```

Python: for statement

Python’s for statement is used to iterate over the items of any sequence like a list, string.

Syntax

```
for_stmt ::=  "for" target_list "in" expression_list ":" suite
              ["else" ":" suite]
```

‘else’ clause is optional, it is executes, once the loop terminates.

test.py

```
names = [ "Phalgun" ,  "Sambith" ,  "Mahesh" ,  "swapna" ]

for name in names:
 print(name)
else:
 print("Exiting from loop")

print ( "Finsihed Execution" )
```

```
$ python3 test.py
Phalgun
Sambith
Mahesh
swapna
Exiting from loop
Finsihed Execution
```

Python: break statement

'break' statement is used to come out of loop like for, while.

test.py

```
i = 0

while (1):
i+=2
 print(i)
 if(i==10):
  break
else:
 print("Exiting loop")

print( "Finished Execution" )
```

```
$ python3 test.py
2
4
6
8
10
Finished Execution
```

As you observe the output, print statement in else clause is not printed. It is because, else clause will not execute, when a break statement terminates the loop.

Python: continue statement

‘continue’ statement skips the current iteration of loops like for, while.

test.py

```
i  =  2

while (i < 20):
i+=2
 if(i%2 != 0):
  continue
 print(i)

else:
 print("Exiting loop")

print( "Finished Execution" )
```

Above program prints all the even numbers up to 20 (exclusive).

```
$ python3 test.py
4
6
8
10
12
14
16
18
20
Exiting loop
Finished Execution
```

Python: functions

A function is a block of statements identified by a name. The keyword 'def' is used to define a function. Functions are mainly used for two reasons.

a. To make code easier to build and understand
b. To reuse the code

Syntax
def functionName(argument1, argument2 …. argumentN):

test.py

```
def factorial(n):
 if(n<=1):
  return 1

result =1

 for i in range(2, n+1):
  result *= i
 return result

print(factorial( 1 ))
print(factorial( 2 ))
print(factorial( 3 ))
print(factorial( 4 ))
print(factorial( 5 ))
```

```
$ python3 test.py
1
2
6
24
120
```

Variables created before function definition may be read inside of the function only if the function does not change the value.
test.py

```
# Create the x variable and set to 44

x  =  44

# Define a simple function that prints x

def f():

x += 1

 print(x)

# Call the function

f()
```

Run above program, you will get following error.

```
$ python3 test.py
Traceback (most recent call last):
  File "test.py", line 11, in <module>
    f()
  File "test.py", line 7, in f
    x += 1
UnboundLocalError: local variable 'x' referenced before assignment
```

Remove ‘x+=1’ statement and re run the above program, value of x is printed to console.

Python: functions: return statement

‘return’ statement is used to return a value from function to the caller.

test.py

```
def factorial(n):
 if(n<=1):
  return 1

result =1

 for i in range(2, n+1):
  result *= i
 return result

print(factorial( 1 ))
print(factorial( 2 ))
print(factorial( 3 ))
print(factorial( 4 ))
print(factorial( 5 ))
```

```
$ python3 test.py
1
2
6
24
120
```

‘return’ without an expression argument returns None.

```
def hello():
	print("Hello")
print(hello())

$ python3 test.py
Hello
None
```

Python lists

List is group of values in between square brackets separated by commas.

```
>>> primes = [ 2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 ]
>>> primes
[2, 3, 5, 7, 11, 13, 17, 19, 23]
```

‘primes’ is a list that contain prime numbers.

```
>>> students = [ "Hari" , "Krishna" , "Kiran" , "Ravi" ]
>>> students
['Hari', 'Krishna', 'Kiran', 'Ravi']
```

‘students’ is a list that contain all student names.

List can contain any type of data.

```
>>> objects = [ 1 , 3 , "Hello" , 10.23 ]
>>> objects
[1, 3, 'Hello', 10.23]
```

You can access elements of list by using index.

```
>>> objects
[1, 3, 'Hello', 10.23]

>>> objects[ 0 ]
1

>>> objects[ 2 ]
'Hello'
```

objects[0] return the first element of list objects.
objects[1] return the second element of list objects.

-ve indexes also used to access elements of a list.

```
>>> objects
```

```
[1, 3, 'Hello', 10.23]
>>> objects[ -1 ]
10.23
>>> objects[ -3 ]
3
```

Slicing
Slicing is used to get sub list.

Example	Description
list[start:end]	Returns sub list from index start (included) to end index (excluded).
list[:end]	Returns sub list from index 0(included) to end index (excluded).
list[start:]	Return sub list from index start to till end.
list[-2:]	Return list from 2nd last to end.

```
>>> objects
[1, 3, 'Hello', 10.23]
>>> objects[ -1 ]
10.23
>>> objects[ -3 ]
3
>>> objects
[1, 3, 'Hello', 10.23]
>>>
>>> objects[:]
```

```
[1, 3, 'Hello', 10.23]
>>>
>>> objects[ 2 :]
['Hello', 10.23]
>>>
>>> objects[: 3 ]
[1, 3, 'Hello']
>>>
>>> objects[ -2 :]
['Hello', 10.23]
```

Concatenate two lists
'+' Operator is used to concatenate two lists.

```
>>> even = [ 2 ,  4 ,  6 ,  8 ,  10 ]
>>> odd = [ 1 ,  3 ,  5 ,  7 ,  9 ]
>>> numbers  =  even + odd
>>> numbers
[2, 4, 6, 8, 10, 1, 3, 5, 7, 9]
```

Lists are mutable; you can change the values of list.

```
>>> numbers
[2, 4, 6, 8, 10, 1, 3, 5, 7, 9]
>>> numbers[ 0 ] =12
>>> numbers[ 1 ] =14
>>> numbers
[12, 14, 6, 8, 10, 1, 3, 5, 7, 9]
```

Add elements to end of list
List provides ‘append’ method to add new elements to the end of a list.

```
>>> numbers
[12, 14, 6, 8, 10, 1, 3, 5, 7, 9]
```

```
>>>
>>> numbers . append( 11 )
>>> numbers . append( 13 )
>>>
>>> numbers
[12, 14, 6, 8, 10, 1, 3, 5, 7, 9, 11, 13]
```

Assignment to slices
If you want to replace sequence of elements in a list, you can use slice notation.

numbers[2:5] = [21, 22, 23]
Above statement replace elements at index 2, 3, 4 with 21, 22, 23 respectively.

numbers[:] = []
Above statement clear the list by replacing all the elements with an empty list.

```
>>> numbers
[12, 14, 6, 8, 10, 1, 3, 5, 7, 9, 11, 13]
>>>
>>> numbers[ 2 : 5 ] = [ 21 ,  22 ,  23 ]
>>> numbers
[12, 14, 21, 22, 23, 1, 3, 5, 7, 9, 11, 13]
>>>
>>> numbers[:]  =  []
>>> numbers
[]
```

Get length of the list
By using ‘len’ function, you can get the length of the list.

```
>>> vowels = [ 'a' ,  'e' ,  'i' ,  'o' ,  'u' ]
>>> len(vowels)
5
```

Nested lists

A list can be nested in other list. For example, in below example, numbers contains 3 lists, first list represent odd numbers, second list represent even numbers and third list represent prime numbers.

```
>>> numbers = [[ 1 ,  3 ,  5 ,  7 ],[ 2 ,  4 ,  6 ,  8 ],[ 2 ,  3 ,  5 ,  7 ,  11 ]]
>>> numbers
[[1, 3, 5, 7], [2, 4, 6, 8], [2, 3, 5, 7, 11]]
>>>
>>> numbers[ 0 ]
[1, 3, 5, 7]
>>>
>>> numbers[ 1 ]
[2, 4, 6, 8]
>>>
>>> numbers[ 2 ]
[2, 3, 5, 7, 11]
>>>

>>>

>>> len(numbers)
3
>>> len(numbers[ 0 ])
4
>>> len(numbers[ 1 ])
4
>>> len(numbers[ 2 ])
5
>>>
>>> numbers[ 0 ][ 1 ] =9
>>> numbers[ 1 ][ 1 : 4 ]  = [ 10 ,  12 ,  14 ]
>>> numbers
[[1, 9, 5, 7], [2, 10, 12, 14], [2, 3, 5, 7, 11]]
```

Python: tuples

A tuple is just like a list, consist of number of values separated by commas.

Differences between tuple and list

a. List is mutable, where as tuple is immutable
b. Tuple can contain heterogeneous data, where as list usually contains homogeneous data.

test.py

```
employee = ( 1 ,  "Hari Krihsna" ,  "Gurram" ,  12345.678 )

print(employee)
print(employee[ 0 ])
print(employee[ 1 ])
print(employee[ 2 ])
print(employee[ 3 ])
```

```
$ python3 test.py
(1, 'Hari Krihsna', 'Gurram', 12345.678)
1
Hari Krihsna
Gurram
12345.678
```

As you observe above example, elements in tuple are enclosed in parenthesis. Eventhough tuples are immutable, you can create tuples which contain mutable objects, such as lists.

test.py

```
employee = ( 1 , [])

print(employee)

employee[1].append(2)
```

```
employee[1].append(4)
employee[1].append(6)

print(employee)
```

```
$ python3 test.py
(1, [])
(1, [2, 4, 6])
```

Packing and unpacking
You can define tuples, without using parenthesis.
For example,
employee=1, "Hari Krihsna", "Gurram", 12345.678

Above one is the example of tuple packing.

id, firstName, lastName, salary = employee
Above one is an example of tuple unpacking. Sequence unpacking requires that there are as many variables on the left side of the equals sign as there are elements in the sequence.

test.py
```
employee =1 ,  "Hari Krihsna" ,  "Gurram" ,  12345.678

id, firstName, lastName, salary  =  employee

print(id)
print(firstName)
print(lastName)
print(salary)
```

```
$ python3 test.py
1
Hari Krihsna
Gurram
12345.678
```

Concatenate tuples
'+' operator is used to concatenate tuples.

```
>>>  tuple1 = ( 1 ,  "HI" ,  2 ,  45.65 )
>>>  tuple2 = ( "abcdef" ,  54 ,  67 )
>>>  tuple3 = tuple1 + tuple2
>>> tuple3
(1, 'HI', 2, 45.65, 'abcdef', 54, 67)
```

Slicing
Just like lists, you can access tuples using slice notation.

Example	**Description**
tuple[start:end]	Returns tuple from index start (included) to end index (excluded).
tuple[:end]	Returns tuple from index 0(included) to end index (excluded).
tuple[start:]	Return tuple from index start to till end.
tuple[-2:]	Return elements from 2nd last to end.

```
>>>  tuple1
(1, 'HI', 2, 45.65)
>>>  tuple1[ 0 :]
(1, 'HI', 2, 45.65)
>>>  tuple1[:]
(1, 'HI', 2, 45.65)
>>>  tuple1[: 3 ]
(1, 'HI', 2)
```

```
>>> tuple1[ 2 : 5 ]
(2, 45.65)
```

‘*’: Repeat tuple elements
‘*’ is the repetition operator, used to repeat the elements of tuple.

```
>>> tuple1
(1, 'HI', 2, 45.65)
>>>
>>> tuple1 *3
(1, 'HI', 2, 45.65, 1, 'HI', 2, 45.65, 1, 'HI', 2, 45.65)
>>>
```

Remove tuple elements
As I said, tuples are immutable, so it is not possible to remove elements from tuple. But you can remove the entire tuple using del statement.

```
>>> tuple1 = ( 1 ,  "HI" ,  2 ,  45.65 )
>>>
>>> del  tuple1
>>> tuple1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError : name 'tuple1' is not defined
```

Observe the output, an exception raised; this is because after deletion, tuple does not exist any more.

Python: Sets

Set is an unordered collection of elements with no duplicates. We can perform union, intersection, difference, and symmetric difference operations on sets.

How to create set

You can create set using {} (or) set() function. 'set()' creates empty set.
>>> evenNumbers={2, 4, 6, 8, 8, 4, 10} >>> evenNumbers {8, 10, 2, 4, 6}

Observe above snippet, evenNumbers is a set that contains even numbers. Observe the output, set doesn't contain duplicate elements.

Following operations are supported by set.

len(s) : cardinality of set

Returns cardinality(Number of distinct elements) of the set.

```
>>> evenNumbers = { 2 , 4 , 6 , 8 , 8 , 4 , 10 }
>>> len(evenNumbers)
5
```

x in s : Check whether element is in set or not

'in' operator is used to check whether element is in set or not, return true if the element is set, else false.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>>
>>> 100 in evenNumbers
False
>>>
>>> 2 in evenNumbers
True
```

x not in s : Check whether element is in set or not

‘not in’ operator is opposite of ‘in’ operator, return true if the element is not in set, else false.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>>
>>> 10 not in evenNumbers
False
>>>
>>> 100 not in evenNumbers
True
```

isdisjoint(other)

Return true if two sets are disjoint, else false. Two sets are said to be disjoint if they have no element in common.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>>
>>> evenNumbers.isdisjoint({1, 3, 5, 7})
True
>>>
>>> evenNumbers.isdisjoint({1, 3, 5, 7, 8})
False
```

issubset(other)

Return true, if this set is subset of other, else false.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>>
>>> evenNumbers.issubset({2, 4})
False
>>>
```

```
>>> evenNumbers . issubset({ 2 ,  4 ,  6 ,  8 ,  10 ,  12 })
True
```

set <= other
Return true if every element in the set is in other.

set < other
Return true, if the set is proper subset of other, that is, set >= other and set != other.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>>
>>> evenNumbers  <= { 2 ,  4 ,  6 ,  8 ,  10 }
True
>>> evenNumbers  <= { 2 ,  4 ,  6 ,  8 ,  10 ,  12 }
True
>>>
>>> evenNumbers  < { 2 ,  4 ,  6 ,  8 ,  10 }
False
>>> evenNumbers  < { 2 ,  4 ,  6 ,  8 ,  10 ,  12 }
True
```

Union of two sets
union(other, ...)
'set | other | ...'

```
>>> evenNumbers = { 2 ,  4 ,  6 ,  8 ,  10 }
>>> oddNumbers = { 1 ,  3 ,  5 ,  7 ,  9 }
>>> result = evenNumbers | oddNumbers
>>> result
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Intersection of two sets
intersection(other, ...)

‘set & other & ...’

```
>>> evenNumbers = { 2 ,  4 ,  6 ,  8 ,  10 }
>>> powersOf2 = { 1 ,  2 ,  4 ,  8 ,  16 }
>>> result = evenNumbers & powersOf2
>>> result
{8, 2, 4}
```

Difference between two sets

difference(other, ...)

‘set - other - ...’

Return a new set with elements in the set that are not in the others.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>> powersOf2
{16, 8, 2, 4, 1}
>>> evenNumbers - powersOf2
{10, 6}
>>> powersOf2 - evenNumbers
{16, 1}
```

Symmetric difference between two sets

symmetric_difference(other)

set ^ other

If A and B are two sets, then Simmetric difference between A and B is A^B = (A-B) union (B-A)

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>> powersOf2
{16, 8, 2, 4, 1}
>>> evenNumbers - powersOf2
{10, 6}
```

```
>>> powersOf2 - evenNumbers
{16, 1}
>>>
>>> evenNumbers ^ powersOf2
{1, 6, 10, 16}
```

Copy elements of set
'copy' function return a new set with a shallow copy of s.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>> temp = evenNumbers . copy()
>>> temp
{8, 10, 2, 4, 6}
```

Update the set
update(other, ...)
set |= other | ...

Update the set by adding elements from other sets.

```
>>> evenNumbers
{8, 10, 2, 4, 6}
>>> oddNumbers
{9, 3, 5, 1, 7}
>>> powersOf2
{16, 8, 2, 4, 1}
>>>
>>> evenNumbers . update(oddNumbers, powersOf2)
>>> evenNumbers
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16}
```

Intersection of all sets
intersection_update(other, ...)

set &= other & ...

Update the set, keeping only elements found in it and all others.

```
>>> numbers
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16}
>>> oddNumbers
{9, 3, 5, 1, 7}
>>> powersOf2
{16, 8, 2, 4, 1}
>>> numbers . intersection_update(oddNumbers, powersOf2)
>>> numbers
{1}
```

Difference update
difference_update(other, ...)
set -= other | ...

Update the set, removing elements found in others.

```
>>> oddNumbers
{9, 3, 5, 1, 7}
>>> powersOf2
{16, 8, 2, 4, 1}
>>> oddNumbers . difference_update(powersOf2)
>>> oddNumbers
{9, 3, 5, 7}
```

Symmetric difference update
symmetric_difference_update
set ^= other
Update the set, keeping only elements found in either set, but not in both.

```
>>> oddNumbers
{9, 3, 5, 7}
```

```
>>> powersOf2
{16, 8, 2, 4, 1}
>>> oddNumbers . symmetric_difference_update(powersOf2)
>>> oddNumbers
{1, 2, 3, 4, 5, 7, 8, 9, 16}
```

Add element to the set
‘add’ method is used to add element to set.

```
>>> temp
{2, 3, 5, 7}
>>>
>>> temp . add( 11 )
>>> temp . add( 13 )
>>>
>>> temp
{2, 3, 5, 7, 11, 13}
```

Remove an element from set
‘remove’ method is used to remove element from set.

```
>>> temp
{2, 3, 5, 7, 11, 13}
>>>
>>> temp . remove( 2 )
>>> temp
{3, 5, 7, 11, 13}
>>>
>>> temp . remove( 11 )
>>> temp
{3, 5, 7, 13}
```

Throws KeyError, if element is not in the set.

```
>>> temp.remove(100)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 100
```

Remove arbitrary element from set
‘pop()’ is used to remove and return an arbitrary element from the set. Throws KeyError, if the set is empty.

```
>>> temp
{5, 7, 13}
>>> temp . pop()
5
>>>
>>> temp . pop()
7
>>> temp . pop()
13
>>> temp . pop()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError : 'pop from an empty set'
```

Remove all elements from set
‘clear’ method is used to remove all elements from set.

```
>>> powersOf2
{16, 8, 2, 4, 1}
>>>
>>> powersOf2 . clear()
>>> powersOf2
set()
```

Python modules

Module is a file, which contains definitions. We can define classes, modules, variables etc., in module file and reuse them in other python scripts.

For example, create a file ‘arithmetic.py’ and copy following code.

arithmetic.py

```
def sum(a, b):
	return a+b

def subtract(a, b):
	return a-b
def mul(a,b):
	return a*b
def div(a, b):
	return a/b
```

Open python interpreter, import the module using ‘import’ key word and call the functions defined in module.

```
>>> import  arithmetic
>>>
>>>  arithmetic . sum( 10 ,  20 )
30
>>>  arithmetic . subtract( 10 ,  20 )
-10
>>>  arithmetic . mul( 10 ,  20 )
200
```

```
>>> arithmetic . div( 10 ,  20 )
0.5
```

How to get the module name

By using the property '__name__', you can get the module name.

```
>>> arithmetic.__name__
'arithmetic
```

Use functions of module as local functions

```
>>> import  arithmetic
>>>  sum = arithmetic . sum
>>>  sub = arithmetic . subtract
>>>  mul = arithmetic . mul
>>>  div = arithmetic . div
>>>
>>>  sum( 10 ,  20 )
30

>>>  mul( 10 ,  20 )
200

>>>  sub( 10 ,  20 )
-10
```

Python command line arguments

Python scripts accept any number of arguments from command line. This option facilitates us to configure Application at the time of running.

How to pass command line arguments
pyhon scriptname arg1 arg2 …argN

By using ‘sys’ module we can access command line arguments.

sys.argv[0] contains file name
sys.argv[1] contains first command line argument, sys.argv[2] contains second command line argument etc.,

arithmetic.py

```
def sum(a, b):
        return a+b

def subtract(a, b):
        return a-b
def mul(a,b):
        return a*b
def div(a, b):
        return a/b
```

main.py

```
import  arithmetic

if __name__ == "__main__":
```

```python
import sys
a = int(sys.argv[1])
b = int(sys.argv[2])

print(sys.argv[0])
print("a = ", a, "b = ", b)
print(arithmetic.sum(a,b))
print(arithmetic.subtract(a,b))
print(arithmetic.mul(a,b))
print(arithmetic.div(a,b))
```

```
$ python3 main.py 10 11
main.py
a =  10 b =  11
21
-1
110
0.9090909090909091

$ python3 main.py 8 13
main.py
a =  8 b =  13
21
-5
104
0.6153846153846154
```

Python: File handling

In this post and subsequent posts, I am going to explain how to open, read and write to file.

How to open file
To read data from a file (or) to write data to a file, we need a reference object that points to file. ‘open’ method is used to get a file object.

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
'open' function opens file in specific mode and return corresponding file object. Throws OSError, if it is unable to open a file.

Parameter	**Description**
file	Full path of the file to be opened.
mode	Specifies the mode, in which the file is opened. By default file opens in read mode.
buffering	0: Switch off the buffer (only allowed in binary mode) 1: Line buffering (only usable in text mode) >1: Specify the size of buffer in bytes. If you don't specify any value, by default buffering works like below.

	a. Binary files are buffered in fixed-size chunks; the size of the buffer is chosen using a heuristic trying to determine the underlying device’s “block size” and falling back on io.DEFAULT_BUFFER_SIZE. b. “Interactive” text files use line buffering.
encoding	Type of encoding used to encode/decode a file. This value should be used in text mode. if encoding is not specified the encoding used is platform dependent.
errors	Specifies how encoding and decoding errors are to be handled. This cannot be used in binary mode
newline	controls how universal newlines mode works. A manner of interpreting text streams in which all of the following are recognized as ending a line: the Unix end-of-line convention '\n', the Windows convention '\r\n', and the old Macintosh convention '\r'.
closefd	If closefd is False and a file descriptor rather than a filename was given, the underlying file descriptor will be kept open when the file is closed. If a filename is given closefd must be

	True (the default) otherwise an error will be raised

Following are different modes that you can use while opening a file.

Mode	Description
'r'	open for reading
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)

For example, data.txt contains following data.
data.txt

```
First line
Second line
Third line

Fourth line
```

```
>>> f = open( "/Users/harikrishna_gurram/data.txt" )
>>>
>>> f . read()
'First line\nSecond line\nThird line\nFourth line\n'
```

Python: classes and objects

Class is a blue print to create objects.

Syntax

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

‘class’ keyword is used to define a class. You can instantiate any number of objects from a class.

Syntax

objName = new ClassName(arguments)

test.py

```
class Employee:
 """ Employee class """
noOfEmployees=0  # Class level variable

 def __init__(self, id, firstName, lastName):
  self.id = id
  self.firstName = firstName
  self.lastName = lastName
  Employee.noOfEmployees = Employee.noOfEmployees + 1
 def displayEmployee(self):
  print(self.id, self.firstName, self.lastName)

emp1 = Employee( 1 , "Hari Krishna" , "Gurram" )
print( "Total Employees" , Employee . noOfEmployees)

emp2 = Employee( 2 , "PTR" , "Nayan" )
print( "Total Employees" , Employee . noOfEmployees)

emp3 = Employee( 3 , "Sankalp" , "Dubey" )
print( "Total Employees" , Employee . noOfEmployees)

emp1.displayEmployee()
emp2.displayEmployee()
emp3.displayEmployee()
```

```
$ python3 test.py
Total Employees 1
Total Employees 2
Total Employees 3
1 Hari Krishna Gurram
2 PTR Nayan
3 Sankalp Dubey
```

__init__(arguments)
__init__ is a special function called constructor used to initialize objects. In Employee class, __init__ method is used to initialize id, firstName, lastName to an object at the time of creation.

noOfEmployees=0
‘noOfEmployees’ is a class variable, shared by all the objects. Class variable are accessed using Class name like ClassName.variableName, ‘Employee.noOfEmployees’ is used to access the class variable noOfEmployees’.

Instance variables
Instance variables have values unique to an object. Usually these are defined in __init__ method. Employee class has 3 instance variables id, firstName, lastName.

The first parameter of any method in a class must be self. This parameter is required even if the function does not use it. ‘self’ is used to refer current object.

Python: Class Vs Instance variables

Class variables are associated with class and available to all the instances (objects) of the class, where as instance variables are unique to objects, used to uniquely identify the object.

Employee.py

```
class Employee:
 """ Blue print for all employees """
 # Class level variables
noOfEmployees=0
organization="abcde corporation"

 def __init__(self, id=-1, firstName="Nil", lastName="Nil"):
  self.id = -1
  self.firstName = firstName
  self.lastName = lastName
  Employee.noOfEmployees+=1

 def displayEmployee(self):
  print(self.id, self.firstName, self.lastName)

emp1 = Employee(id=1, firstName="Hari Krishna",
lastName="Gurram")
emp1.displayEmployee()
print( "Total Employees : " , Employee . noOfEmployees)
print( "Organization : " , Employee . organization)
```

```
emp2 = Employee(id=3, firstName="PTR")
emp2.displayEmployee()
print( "Total Employees : " , Employee . noOfEmployees)
print( "Organization : " , Employee . organization)
```

```
$ python3 Employee.py
-1 Hari Krishna Gurram
Total Employees :  1
Organization :  abcde corporation
-1 PTR Nil
Total Employees :  2
Organization :  abcde corporation
```

As you observe Employee.py noOfEmployees, organization are class variables, are available to all the instances. Class variables are accessed using ClassName followed by dot followed by variable name.

Employee.noOfEmployees: is used to access class variable noOfEmployees.
Employee.organization: is used to access class variable organization.

Python: Inheritance

Inheritance is the concept of re usability. Object of one class can get the properties and methods of object of another class by using inheritance.

Syntax

```
class DerivedClassName(BaseClassName1, BaseClassName2 ... BaseClassNameN):
    <statement-1>
    .
    .
    .
    <statement-N>
```

inheritance.py

```
class Parent:
 def printLastName(self):
  print("Gurram")

 def printPermAddress(self):
  print("State : Andhra Pradesh")
  print("Country : India")

class Child(Parent):
 def printName(self):
  print("Hari Krishna Gurram")
```

```
child1  =  Child()

child1.printName()
child1.printLastName()
child1.printPermAddress()
```

```
$ python3 inheritance.py
Hari Krishna Gurram
Gurram
State : Andhra Pradesh
Country : India
```

Observe above program, two classes parent and child are defined. Parent class defines two methods printLastName, printPermAddress.
Child class defines one method printName. Child class inheriting the methods printLastName, printPermAddress from parent class.

Overriding the methods of Parent class
You can override the properties, methods of parent class in child class. For example, following application overrides ‘printPermAddress’ method of Parent class.

inheritance.py

```
class Parent:
 def printLastName(self):
  print("Gurram")

 def printPermAddress(self):
  print("State : Andhra Pradesh")
  print("Country : India")
```

```
class Child(Parent):
 def printName(self):
  print("Hari Krishna")

 def printPermAddress(self):
  print("City : Bangalore")
  print("State : Karnataka")
  print("Country : India")

child1  =  Child()

child1.printName()
child1.printLastName()
child1.printPermAddress()
```

```
$ python3 inheritance.py
Hari Krishna
Gurram
City : Bangalore
State : Karnataka
Country : India
```

Python: Exceptions

Exception is an event that disrupts the normal flow of execution. Even though statements in your program are syntactically correct, they may cause an error.

```
>>> 10/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError : division by zero
>>>
>>>  tempList = []
>>>  tempList[ 20 ]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError : list index out of range
```

Observer above snippet, ‘10/0’ causes ZeroDivisionError. When program tries to access 20th element of tempList it causes IndexError.

Python: Handling Exceptions

Python provide keywords try, except to handle exceptions.

test.py

```
while True:
 try:
  x = int(input("Enter input "))
  print(x)
  break;
 except ValueError:
  print("Please enter valid number")
```

```
$ python3 test.py
Enter input an
Please enter valid number
Enter input ana
Please enter valid number
Enter input ptr
Please enter valid number
Enter input 10
10
```

How try and except block work?

The statements in try block executed first. If no exception occurs, the except clauses are skipped and execution of the try statement is finished. If any exception occurs during the execution of try block, the rest of the try clause is skipped.

‘try’ block followed by number of except clauses, if exception thrown in try cause matches to any exception followed by except clause, then particular except clause is executed.

If an exception occurs which does not match the exception named in the except clause, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops by throwing exception message.

Try block can be followed by multiple except clauses

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))
  print(x/y)
  break;
 except ValueError:
  print("Please enter valid number")
 except ZeroDivisionError:
  print("y should be non zero")
```

```
$ python3 test.py
Enter divisor 2
Enter dividend 0
y should be non zero
Enter divisor 4
Enter dividend 0
y should be non zero
Enter divisor 4
Enter dividend 2
2.0
```

Handling multiple exceptions in single except clause
An except clause can catch more than one exception. For example ‘except (ValueError, ZeroDivisionError)’ can handle both ValueError and ZeroDivisionError.

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))
  print(x/y)
  break;
 except (ValueError, ZeroDivisionError):
  print("Please enter valid number (or) y should be greater than 0")
```

```
$ python3 test.py
Enter divisor 2
Enter dividend 0
Please enter valid number (or) y should be greater than 0
Enter divisor aa
Please enter valid number (or) y should be greater than 0
Enter divisor 2
Enter dividend 4
0.5
```

Last except clause can omit exception name. It is used as global exception handler.

test.py

```
while True:
 try:
  tempList=[]
  print(tempList[10])
  break
 except ValueError:
  print("Please enter valid number")
 except ZeroDivisionError:
  print("y should be non zero")
 except Exception as inst:
  print("Global handler", inst)
  break
```

```
$ python3 test.py
Global handler list index out of range
```

try…except…else clause

'try…except' statement can have optional else clause, it is followed by except clause. If try block doesn't throw any exception, else clause will be executed.

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))
  print(x/y)
 except ValueError:
  print("Please enter valid number")
 except ZeroDivisionError:
  print("y should be non zero")
 else:
  print("Program executed successfully")
  break
```

```
$ python3 test.py
Enter divisor 4
Enter dividend 2
2.0
Program executed successfully
```

Exception argument
Whenever an exception occurs, it is associated with a variable called exception argument.

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))
  print(x/y)
 except ValueError as inst:
  print(inst)
 except ZeroDivisionError as inst:
  print(inst)
 else:
  print("Program executed successfully")
  break
```

```
$ python3 test.py
Enter divisor qwerty
invalid literal for int() with base 10: 'qwerty'
Enter divisor 4
Enter dividend 0
division by zero
Enter divisor 2
Enter dividend 4
0.5
Program executed successfully
```

The except clause can specify a variable after the exception name. The variable is bound to an exception instance with the arguments stored in instance.args.

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))

  if y==0:
   raise Exception(x, y)
  print("x/y = ",x/y)
  break
 except Exception as inst:
  arg1, arg2 = inst.args
  print("arg1=", arg1)
  print("arg2=", arg2)
```

```
$ python3 test.py
Enter divisor 2
Enter dividend 0
arg1= 2
arg2= 0
Enter divisor 2
Enter dividend 4
x/y =  0.5
```

If an exception has arguments associated with it, those are printed as last part of the exception message.

test.py

```
while True:
 try:
  x = int(input("Enter divisor "))
  y = int(input("Enter dividend "))

  if y==0:
   raise Exception(x, y)
  print("x/y = ",x/y)
  break
 except Exception as inst:
  print(inst)
```

```
$ python3 test.py
Enter divisor 2
Enter dividend 0
(2, 0)
Enter divisor 2
Enter dividend 4
x/y =  0.5
```

Python global keyword

‘global’ keyword is used to create/change a global variables (You can declare functions, classes etc. also) from local context.

test.py

```
def function1():
 global data
 data="Hello World"

def function2():
 print(data)

function1()
function2()
```

```
$ python3 test.py
Hello World
```

Observe ‘test.py’, even though data is declared in function1, it is accessed by function2. It is because, data is declared in global scope.

Python: Get type of variable

You can get the type of a variable using 'type()' function or __class__ property.

```
>>> data = [ 1 ,  2 ,  3 ,  4 ]
>>> type(data)
<class 'list'>

>>> data . __class__
<class 'list'>

>>>
>>> data = { 1 : "hari" ,  2 : "Krishna" }
>>> type(data)
<class 'dict'>

>>> data . __class__
<class 'dict'>
```

Checking the type using if statement

```
data = { 1 : "hari" ,  2 : "Krishna" }

#Approach 1

if(data.__class__.__name__ == 'dict' ):

   print("data is of type dictionary")

else:

   print("data is not dictionary type")


#Approach 2

if(type(data).__name__ == 'dict'):

   print("data is of type dictionary")

else:

   print("data is not dictionary type")
```

```
#Approach 3
if type(data)==type(dict()):
    print("data is of type dictionary")
else:
    print("data is not dictionary type")
```

Run above program, you will get following output.

```
data is of type dictionary
data is of type dictionary
data is of type dictionary
```

Check whether an object is instance of class or not
‘isinstance(object, classinfo)’ method is used to check whether an object is instance of given class or not. Return true if the object argument is an instance of the classinfo argument, or of a subclass thereof, else false.

```
data = { 1 : "hari" ,  2 : "Krishna" }

class Employee:
    def __init__(self, id, firstName, lastName):
     self.id = id
     self.firstName = firstName
     self.lastName = lastName

emp = Employee( 1 ,  "Hari" ,  "Krishna" )

print(isinstance(emp, Employee))
print(isinstance(emp, dict))
print(isinstance(data, dict))
```

Run above program, you will get following output.

```
True
False
True
```

CODING EXERCISES

PYTHON, JAVA AND C#

CODING FOR BEGINNERS

JJ TAM

PYTHON CODING EXERCISES

CODING FOR BEGINNERS

JJ TAM

Python Basic – Exercises

Get Python version

CODE

```
import sys
print("Python version")
print (sys.version)
print("Version info.")
print (sys.version_info)
```

OUTPUT

```
Python version

3.6.6 (default, Jun 28 2018, 04:42:43)

[GCC 5.4.0 20160609]

Version info.

sys.version_info(major=3, minor=6, micro=6, releaselevel='final',
serial=0)
```

Display current date and time

PYTHON CODE

```
import datetime
now = datetime.datetime.now()
print ("Current date and time : ")
print (now.strftime("%Y-%m-%d %H:%M:%S"))
```

OUTPUT

```
Current date and time :

2020-10-22 10:35:31
```

Print the calendar

PYTHON CODE

```
import calendar

y = int(input("Input the year : "))

m = int(input("Input the month : "))

print(calendar.month(y, m))
```

OUTPUT

```
Input the year : 2020
Input the month : 10

    October 2020
Mo Tu We Th Fr Sa Su
          1  2  3  4
5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Computes the value of n+nn+nnn

PYTHON CODE

```
a = int(input("Input an integer : "))

n1 = int( "%s" % a )

n2 = int( "%s%s" % (a,a) )

n3 = int( "%s%s%s" % (a,a,a) )

print (n1+n2+n3)
```

OUTPUT

```
Input an integer : 6

738
```

Calculate number of days

PROGRAM

```
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2015, 7, 11)
delta = l_date - f_date
print(delta.days)
```

OUTPUT

```
374
```

volume of a sphere in Python

PROGRAM

```
pi = 3.1415926535897931
r= 6.0
V= 4.0/3.0*pi* r**3
print('The volume of the sphere is: ',V)
```

OUTPUT

```
The volume of the sphere is:  904.7786842338603
```

Compute the area of Triangle

PROGRAM

```
b = int(input("Input the base : "))
h = int(input("Input the height : "))

area = b*h/2

print("area = ", area)
```

OUTPUT

```
Input the base : 20

Input the height : 40

area =  400.0
```

Compute the GCD

PROGRAM

```
def gcd(x, y):
    gcd = 1

    if x % y == 0:
        return y

    for k in range(int(y / 2), 0, -1):
        if x % k == 0 and y % k == 0:
            gcd = k
            break
    return gcd

print(gcd(12, 17))
print(gcd(4, 6))
```

OUTPUT

```
1

2
```

CALCULATE THE LCM

PROGRAM

```
def lcm(x, y):
   if x > y:
       z = x
   else:
       z = y

   while(True):
       if((z % x == 0) and (z % y == 0)):
           lcm = z
           break
       z += 1

   return lcm
print(lcm(4, 6))
print(lcm(15, 17))
```

OUTPUT

```
12

255
```

Convert feet and inches to centimeters

PROGRAM

```
print("Input your height: ")
h_ft = int(input("Feet: "))
h_inch = int(input("Inches: "))

h_inch += h_ft * 12
h_cm = round(h_inch * 2.54, 1)

print("Your height is : %d cm." % h_cm)
```

OUTPUT

```
Input your height:

Feet:  5

Inches:  3

Your height is : 160 cm.
```

Convert time – seconds

PYTHON CODE

```
days = int(input("Input days: ")) * 3600 * 24
hours = int(input("Input hours: ")) * 3600
minutes = int(input("Input minutes: ")) * 60
seconds = int(input("Input seconds: "))

time = days + hours + minutes + seconds

print("The  amounts of seconds", time)
```

Output:

```
Input days:  4

Input hours:  5

Input minutes:  20

Input seconds:  10

The  amounts of seconds 364810
```

Convert seconds to day

PROGRAM

```
time = float(input("Input time in seconds: "))
day = time // (24 * 3600)
time = time % (24 * 3600)
hour = time // 3600
time %= 3600
minutes = time // 60
time %= 60
seconds = time
print("d:h:m:s-> %d:%d:%d:%d" % (day, hour, minutes, seconds))
```

OUTPUT

```
Input time in seconds:
1234565

d:h:m:s-> 14:6:56:5
```

Calculate BMS

PROGRAM

```
height = float(input("Input your height in meters: "))
weight = float(input("Input your weight in kilogram: "))
print("Your body mass index is: ", round(weight / (height * height), 2))
```

OUTPUT

```
Input your height in meters: 6.2

Input your weight in kilogram: 72

Your body mass index is:  1.87
```

Sort three integers

PROGRAM

```
x = int(input("Input first number: "))
y = int(input("Input second number: "))
z = int(input("Input third number: "))

a1 = min(x, y, z)
a3 = max(x, y, z)
a2 = (x + y + z) - a1 - a3
print("Numbers in sorted order: ", a1, a2, a3)
```

OUTPUT

```
Input first number:
2
Input second number:
4
Input third number:
5
Numbers in sorted order:  2 4 5
```

Get system time

PROGRAM

```
import time
print()
print(time.ctime())
print()
```

OUTPUT

```
Thu Oct 22 14:59:27 2020
```

Check a number

PYTHON CODE

```
num = float(input("Input a number: "))
if num > 0:
   print("It is positive number")
elif num == 0:
   print("It is Zero")
else:
   print("It is a negative number")
```

OUTPUT

```
Input a number: 200

It is positive number
```

Python code to Remove first item

PROGRAM

```
color = ["Red", "Black", "Green", "White", "Orange"]
print("\nOriginal Color: ",color)
del color[0]
print("After removing the first color: ",color)
print()
```

OUTPUT

```
Original Color:  ['Red', 'Black', 'Green', 'White', 'Orange']

After removing the first color:  ['Black', 'Green', 'White', 'Orange']
```

Filter positive numbers

PYTHON CODE

```
nums = [34, 1, 0, -23]
print("Original numbers in the list: ",nums)
new_nums = list(filter(lambda x: x >0, nums))
print("Positive numbers in the list: ",new_nums)
```

OUTPUT

```
Original numbers in the list:  [34, 1, 0,
-23]

Positive numbers in the list:  [34, 1]
```

Count the number 4

in a given list

SAMPLE PROGRAM

```
def list_count_4(nums):
  count = 0
  for num in nums:
    if num == 4:
      count = count + 1

  return count

print(list_count_4([1, 4, 6, 7, 4]))
print(list_count_4([1, 4, 6, 4, 7, 4]))
```

OUTPUT

```
2


3
```

Find a number even or odd

SAMPLE PROGRAM

```
num = int(input("Enter a number: "))
mod = num % 2
if mod > 0:
    print("This is an odd number.")
else:
    print("This is an even number.")
```

OUTPUT

```
Enter a number:
5

This is an odd number.
```

Difference between the two lists

PYTHON CODE

```
list1 = [1, 3, 5, 7, 9]
list2=[1, 2, 4, 6, 7, 8]
diff_list1_list2 = list(set(list1) - set(list2))
diff_list2_list1 = list(set(list2) - set(list1))
total_diff = diff_list1_list2 + diff_list2_list1
print(total_diff)
```

OUTPUT

```
[9, 3, 5, 8, 2, 4, 6]
```

Generate all permutations

of a list

PYTHON CODE

```
import itertools
print(list(itertools.permutations([1,2,3])))
```

OUTPUT

```
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

Find the second smallest number in a list

PYTHON PROGRAM

```
def second_smallest(numbers):
  if (len(numbers)<2):
    return
  if ((len(numbers)==2)  and (numbers[0] == numbers[1]) ):
    return
  dup_items = set()
  uniq_items = []
  for x in numbers:
    if x not in dup_items:
      uniq_items.append(x)
      dup_items.add(x)
  uniq_items.sort()
  return  uniq_items[1]

print(second_smallest([1, 2, -8, -2, 0, -2]))
print(second_smallest([1, 1, 0, 0, 2, -2, -2]))
print(second_smallest([1, 1, 1, 0, 0, 0, 2, -2, -2]))
print(second_smallest([2,2]))
print(second_smallest([2]))
```

OUTPUT

```
-2
```

0

0

None

None

Get unique values

from a list

Python Code

```
my_list = [10, 20, 30, 40, 20, 50, 60, 40]
print("Original List : ",my_list)
my_set = set(my_list)
my_new_list = list(my_set)
print("List of unique numbers : ",my_new_list)
```

OUTPUT

```
Original List :  [10, 20, 30, 40, 20, 50, 60,
40]
List of unique numbers :  [40, 10, 50, 20, 60, 30]
```

Get the frequency of the elements

Python Code

```
import collections
my_list = [10,10,10,10,20,20,20,20,40,40,50,50,30]
print("Original List : ",my_list)
ctr = collections.Counter(my_list)
print("Frequency of the elements in the List : ",ctr)
```

OUTPUT

```
Original List :  [10, 10, 10, 10, 20, 20, 20, 20, 40, 40, 50, 50,
30]

Frequency of the elements in the List :  Counter({10: 4, 20: 4, 40: 2,
50: 2, 30: 1})
```

Generate all sublists

of a list in Python

Python Code

```
from itertools import combinations
def sub_lists(my_list):
        subs = []
        for i in range(0, len(my_list)+1):
          temp = [list(x) for x in combinations(my_list, i)]
          if len(temp)>0:
            subs.extend(temp)
        return subs


l1 = [10, 20, 30, 40]
l2 = ['X', 'Y', 'Z']
print("Original list:")
print(l1)
print("S")
print(sub_lists(l1))
print("Sublists of the said list:")
print(sub_lists(l1))
print("\nOriginal list:")
print(l2)
print("Sublists of the said list:")
print(sub_lists(l2))
```

OUTPUT

```
Original list:

[10, 20, 30, 40]

S

[[], [10], [20], [30], [40], [10, 20], [10, 30], [10, 40], [20, 30], [20, 40],
[30, 40], [10, 20, 30], [10, 20, 40], [10, 30, 40], [20, 30, 40], [10, 20,
30, 40]]

Sublists of the said list:

[[], [10], [20], [30], [40], [10, 20], [10, 30], [10, 40], [20, 30], [20, 40],
[30, 40], [10, 20, 30], [10, 20, 40], [10, 30, 40], [20, 30, 40], [10, 20,
30, 40]]


Original list:

['X', 'Y', 'Z']

Sublists of the said list:

[[], ['X'], ['Y'], ['Z'], ['X', 'Y'], ['X', 'Z'], ['Y', 'Z'], ['X', 'Y', 'Z']]
```

Find common items

from two lists

Python Code

```
color1 = "Red", "Green", "Orange", "White"
color2 = "Black", "Green", "White", "Pink"
print(set(color1) & set(color2))
```

OUTPUT

```
'Green', 'White'}
```

Create a list

with infinite elements

Python Code

```
import itertools
c = itertools.count()
print(next(c))
print(next(c))
print(next(c))
print(next(c))
print(next(c))
```

OUTPUT

```
0

1

2

3

4
```

Remove consecutive duplicates of a given list

Python Code

```
from itertools import groupby
def compress(l_nums):
   return [key for key, group in groupby(l_nums)]
n_list = [0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4 ]
print("Original list:")
print(n_list)
print("\nAfter removing consecutive duplicates:")
print(compress(n_list))
```

OUTPUT

```
Original list:
[0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4]

After removing consecutive duplicates:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 4]
```

Flatten a nested

list structure

Python Code

```
def flatten_list(n_list):
    result_list = []
    if not n_list: return result_list
    stack = [list(n_list)]
    while stack:
        c_num = stack.pop()
        next = c_num.pop()
        if c_num: stack.append(c_num)
        if isinstance(next, list):
            if next: stack.append(list(next))
        else: result_list.append(next)
    result_list.reverse()
    return result_list
n_list = [0, 10, [20, 30], 40, 50, [60, 70, 80], [90, 100, 110, 120]]
print("Original list:")
print(n_list)
print("\nFlatten list:")
print(flatten_list(n_list))
```

OUTPUT

```
Original list:

[0, 10, [20, 30], 40, 50, [60, 70, 80], [90, 100, 110, 120]]


Flatten list:

[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
```

JAVA
CODING EXERCISES

CODING FOR BEGINNERS

JJ TAM

JAVA CODING EXERCISES

Print hello

And your name

On a separate lines

JAVACODE

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args)
  {
    Scanner input = new Scanner (System.in);
    System.out.print("Input your first name: ");
    String fname = input.next();
    System.out.print("Input your last name: ");
    String lname = input.next();
    System.out.println();
    System.out.println("Hello \n"+fname+" "+lname);
  }
}
```

OUTPUT

```
Input your first name: JJ
Input your last name: TAM


Hello

JJ TAM
```

Print the sum

Of two numbers

JAVACODE

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args)
  {
    Scanner input = new Scanner (System.in);
    System.out.print("Input the first number: ");
    int num1 = input.nextInt();
    System.out.print("Input the second number: ");
    int num2 = input.nextInt();
    int sum = num1 + num2;
    System.out.println();
    System.out.println("Sum: "+sum);
  }
}
```

OUTPUT

```
Input the first number: 545

Input the second number: 455
```

Sum: 1000

Divide two numbers

And print on the screen

JAVA CODE

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args)
  {
    Scanner input = new Scanner (System.in);
    System.out.print("Input the first number: ");
    int a = input.nextInt();
    System.out.print("Input the second number: ");
    int b = input.nextInt();
    int d = (a/b);
    System.out.println();
    System.out.println("The division of a and b is:" +d);
  }
}
```

OUTPUT

```
Input the first number: 100

Input the second number: 25
```

```
The division of a and b is:4
```

Print the result

Of specified operations

JAVA CODE

```
public class Main {

public static void main(String[] args) {
   int w = -5 + 8 * 6;
   int x = (55 + 9) % 9;
   int y = 20 + (-3 * 5 / 8);
   int z = 5 + 15 / 3 * 2 - 8 % 3;

   System.out.print(w + "\n" + x + "\n" + y + "\n" + z);
  }
}
```

Print multiplication table

JAVA CODE

```
import java.util.Scanner;

public class Main {

public static void main(String[] args) {
  Scanner in = new Scanner(System.in);
  System.out.println("Input the Number: ");
  int n = in .nextInt();
  for (int i = 1; i <= 10; i++) {
   System.out.println(n + "*" + i + " = " + (n * i));
  }
}
}
```

OUTPUT

```
Input the Number:

15

15*1 = 15

15*2 = 30

15*3 = 45

15*4 = 60
```

15*5 = 75

15*6 = 90

15*7 = 105

15*8 = 120

15*9 = 135

15*10 = 150

Find the area

And perimeter of a circle

JAVA CODE

```
import java.util.Scanner;

public class Main {

public static void main(String[] args) {

  Scanner io = new Scanner(System.in);

  System.out.println("Input the radius of the circle: ");

  double radius = io.nextDouble();

  System.out.println("Perimeter is = " + (2 * radius * Math.PI));

  System.out.println("Area is = " + (Math.PI * radius * radius));

}

}
```

OUTPUT

```
Input the radius of the circle:

10

Perimeter is = 62.83185307179586

Area is = 314.1592653589793
```

Swap two variables

JAVA CODE

```
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
   int x, y, z;
   Scanner in = new Scanner(System.in);

   System.out.println("Input the first number: ");
   x = in.nextInt();
   System.out.println("Input the second number: ");
   y = in.nextInt();

   z = x;
   x = y;
   y = z;

   System.out.println(" Swapped values are3:" + x + " and " + y);
  }
}
```

OUTPUT

```
Input the first number:

25

Input the second number:
```

```
50
Swapped values are3:50 and 25
```

Print a face

JAVA CODE

```
public class Main {

public static void main(String[] args) {

  String[] arra = new String[5];

  arra[0] = " +\"\"\"\"\"+ ";
  arra[1] = "[| o o |]";
  arra[2] = " |  ^  |";
  arra[3] = " | '-' |";
  arra[4] = " +-----+";

  for (int i = 0; i < 5; i++) {
   System.out.println(arra[i]);
  }

}

}
```

OUTPUT

```
+"""""+

[| o o |]
```

```
|  ^  |
| '-' |
+-----+
```

Fahrenheit to Celsius degree

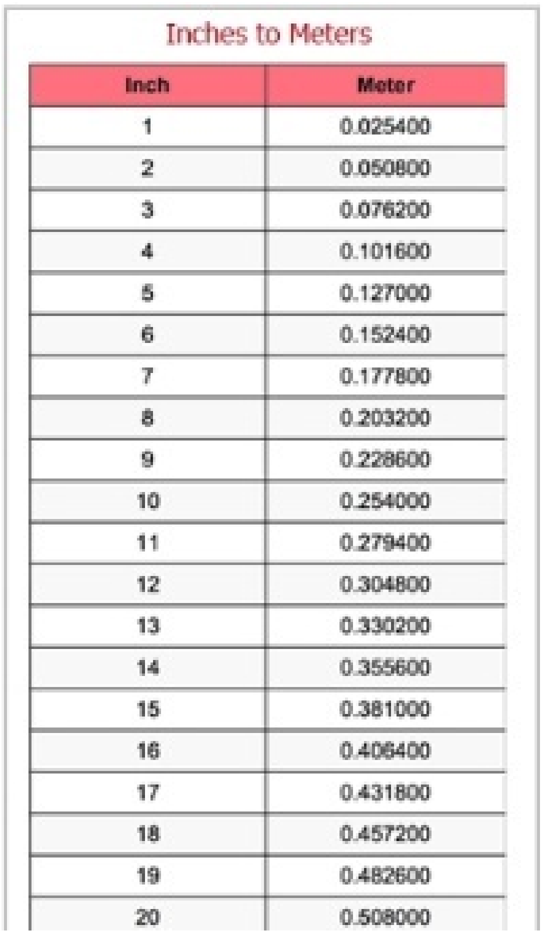
JAVA CODE

```
import java.util.Scanner;
public class Main {

    public static void main(String[] Strings) {

        Scanner input = new Scanner(System.in);

        System.out.print("Input a degree in Fahrenheit: ");
        double fahrenheit = input.nextDouble();

        double  celsius =(( 5 *(fahrenheit - 32.0)) / 9.0);
        System.out.println(fahrenheit + " degree Fahrenheit is equal to "
+ celsius + " in Celsius");
    }
}
```

OUTPUT

```
Input a degree in Fahrenheit: 96

96.0 degree Fahrenheit is equal to 35.55555555555556 in Celsius
```

Inches to meters



Inches to Meters

Inch	Meter
1	0.025400
2	0.050800
3	0.076200
4	0.101600
5	0.127000
6	0.152400
7	0.177800
8	0.203200
9	0.228600
10	0.254000
11	0.279400
12	0.304800
13	0.330200
14	0.355600
15	0.381000
16	0.406400
17	0.431800
18	0.457200
19	0.482600
20	0.508000

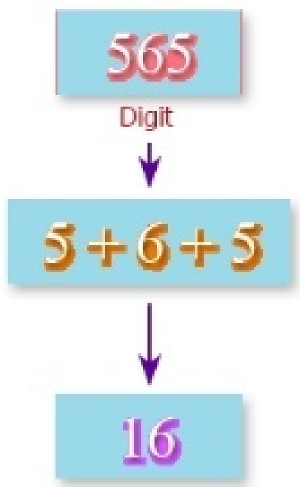
JAVA CODE

```
import java.util.Scanner;
public class Main {

    public static void main(String[] Strings) {

        Scanner input = new Scanner(System.in);

        System.out.print("Input a value for inch: ");
        double inch = input.nextDouble();
        double meters = inch * 0.0254;
        System.out.println(inch + " inch is " + meters + " meters");

    }
}
```

OUTPUT

```
Input a value for inch: 5

5.0 inch is 0.127 meters
```

Adds all the digits



JAVA CODE

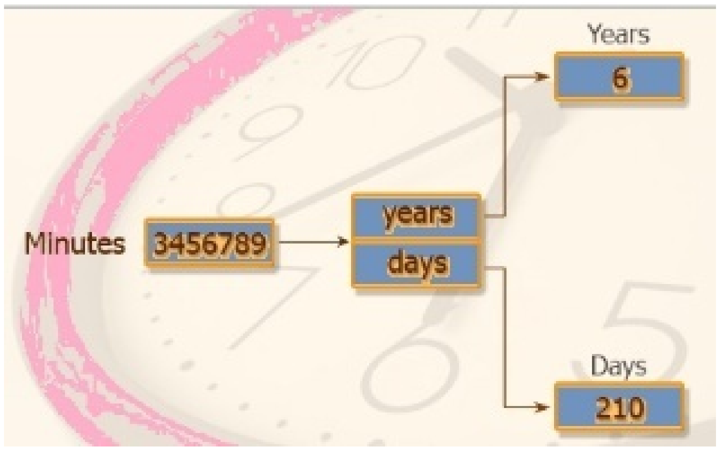
```
import java.util.Scanner;
public class Main {

    public static void main(String[] Strings) {

        Scanner input = new Scanner(System.in);

        System.out.print("Input an integer between 0 and 1000: ");
        int num = input.nextInt();

        int firstDigit = num % 10;
        int remainingNumber = num / 10;
        int SecondDigit = remainingNumber % 10;
        remainingNumber = remainingNumber / 10;
        int thirdDigit = remainingNumber % 10;
        remainingNumber = remainingNumber / 10;
        int fourthDigit = remainingNumber % 10;
        int sum = thirdDigit + SecondDigit + firstDigit + fourthDigit;
        System.out.println("The sum of all digits in " + num + " is " +
sum);

    }
}
```

OUTPUT

```
Input an integer between 0 and 1000: 750

The sum of all digits in 750 is 12
```

Print the number of years and days

From minutes



JAVA CODE

```
import java.util.Scanner;
public class Main {

    public static void main(String[] Strings) {


        double minutesInYear = 60 * 24 * 365;

        Scanner input = new Scanner(System.in);

        System.out.print("Input the number of minutes: ");
        double min = input.nextDouble();

        long years = (long) (min / minutesInYear);
        int days = (int) (min / 60 / 24) % 365;

        System.out.println((int) min + " minutes is approximately " +
years + " years and " + days + " days");
    }
}
```

OUTPUT

```
Input the number of minutes: 123456

123456 minutes is approximately 0 years and 85 days
```

Compute (BMI)

Body mass index

JAVA CODE

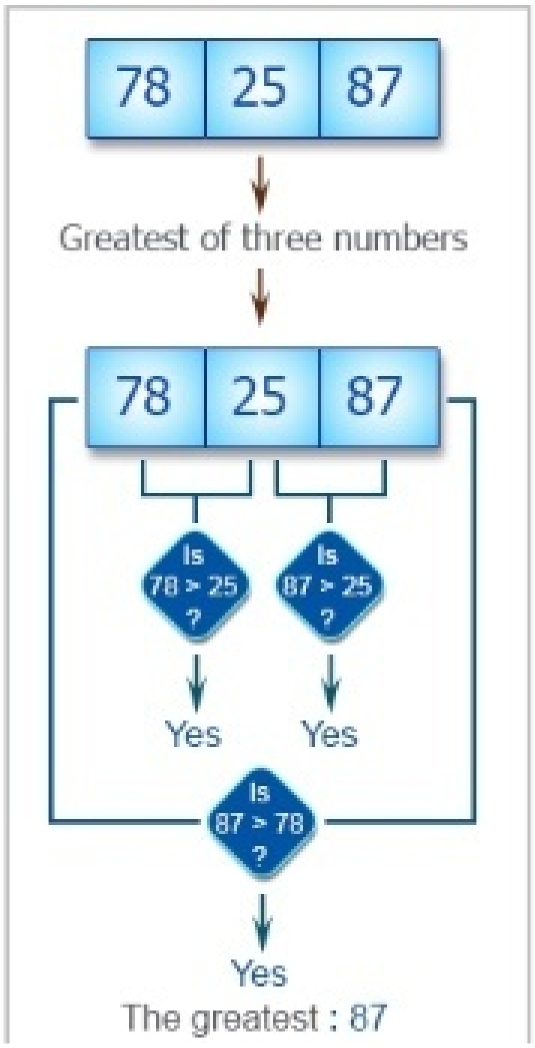
```
import java.util.Scanner;
public class Main {

    public static void main(String[] Strings) {

        Scanner input = new Scanner(System.in);

        System.out.print("Input weight in pounds: ");
        double weight = input.nextDouble();

        System.out.print("Input height in inches: ");
        double inches = input.nextDouble();

        double BMI = weight * 0.45359237 / (inches * 0.0254 * inches *
0.0254);
        System.out.print("Body Mass Index is " + BMI+"\n");
    }
}
```

OUTPUT

```
Input weight in pounds: 450
Input height in inches: 72
Body Mass Index is 61.03034545478814
```

Find the greatest of three numbers

Java Conditional Statement



JAVA CODE

```
import java.util.Scanner;
public class Main {
}



  public static void main(String[] args) {
  Scanner in = new Scanner(System.in);

  System.out.print("Input the 1st number: ");
  int num1 = in.nextInt();

  System.out.print("Input the 2nd number: ");
  int num2 = in.nextInt();

  System.out.print("Input the 3rd number: ");
  int num3 = in.nextInt();


  if (num1 > num2)
   if (num1 > num3)
    System.out.println("The greatest: " + num1);

  if (num2 > num1)
   if (num2 > num3)
    System.out.println("The greatest: " + num2);

  if (num3 > num1)
   if (num3 > num2)
    System.out.println("The greatest: " + num3);
}
}
```

OUTPUT

```
Input the 1st number: 25
Input the 2nd number: 58
Input the 3rd number: 98
The greatest: 98
```

Display the multiplication table

of a given integer

JAVA CODE

```
import java.util.Scanner;
public class Main {

   public static void main(String[] args)

{
   int j,n;

   System.out.print("Input the number(Table to be calculated): ");
{
   System.out.print("Input number of terms : ");
    Scanner in = new Scanner(System.in);
            n = in.nextInt();

   System.out.println ("\n");
   for(j=0;j<=n;j++)

     System.out.println(n+" X "+j+" = " +n*j);
   }
}
}
```

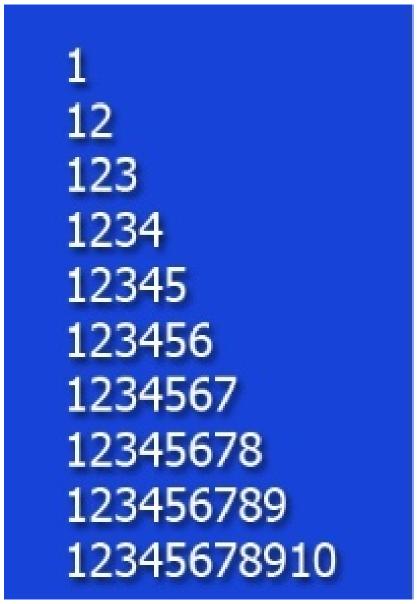
OUTPUT

```
Input the number(Table to be calculated): Input number of terms : 7


7 X 0 = 0

7 X 1 = 7

7 X 2 = 14

7 X 3 = 21

7 X 4 = 28

7 X 5 = 35

7 X 6 = 42

7 X 7 = 49
```

Display the pattern

like right angle triangle

with a number



```
1
12
123
1234
12345
123456
1234567
12345678
123456789
12345678910
```

JAVA CODE

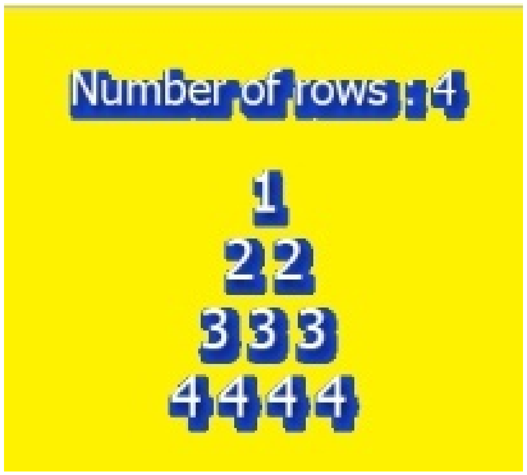
```
import java.util.Scanner;
public class Main {

   public static void main(String[] args)

{
   int i,j,n;
   System.out.print("Input number of rows : ");
Scanner in = new Scanner(System.in);
```

```
            n = in.nextInt();

   for(i=1;i<=n;i++)
   {
        for(j=1;j<=i;j++)
          System.out.print(j);

    System.out.println("");
    }
}
}
```

OUTPUT

```
Input number of rows : 9
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

Print a pattern like a pyramid


```
Number of rows : 4

   1
  2 2
 3 3 3
4 4 4 4
```

JAVA CODE

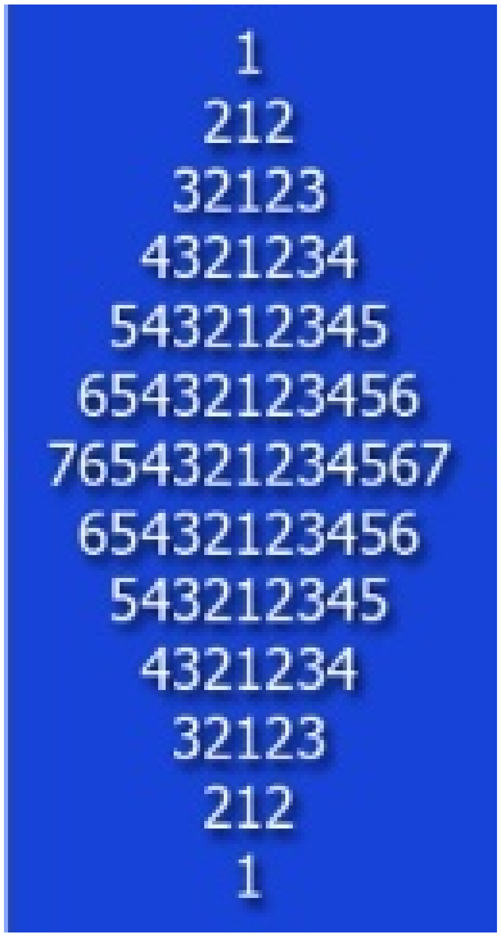
```
import java.util.Scanner;
public class Main {


  public static void main(String[] args)
{
   int i,j,n,s,x;
   System.out.print ("Input number of rows : ");
   Scanner in = new Scanner(System.in);
            n = in.nextInt();


   s=n+4-1;
    for(i=1;i<=n;i++)
  {
  for(x=s;x!=0;x--)
   {
  System.out.print(" ");
   }
   for(j=1;j<=i;j++)
   {
    System.out.print(i+" ");
    }
        System.out.println();
   s--;
  }
}
}
```

OUTPUT

```
Input number of rows : 7
          1
         2 2
        3 3 3
       4 4 4 4
      5 5 5 5 5
     6 6 6 6 6 6
```

7 7 7 7 7 7 7

Display the number rhombus structure



```
      1
     212
    32123
   4321234
  543212345
 65432123456
7654321234567
 65432123456
  543212345
   4321234
    32123
     212
      1
```

JAVA CODE

```
import java.util.Scanner;
public class Main {
  public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);
        System.out.print("Input the number:  ");
```

```
int n = in.nextInt();
int count = 1;
int no_of_spaces = 1;
int start = 0;

for (int i = 1; i < (n * 2); i++)
{

    for (int spc = n - no_of_spaces; spc > 0; spc--)
    {
        System.out.print(" ");
    }
    if (i < n)
    {
        start = i;          //for number
        no_of_spaces++;    //for spaces
    } else
    {
        start = n * 2 - i;   //for number
        no_of_spaces--;      //for space
    }
    for (int j = 0; j < count; j++)
    {
        System.out.print(start);
        if (j < count / 2)
        {
            start--;
        } else
```

```
                {
                    start++;
                }
            }
            if (i < n)
            {
                count = count + 2;
            } else {
                count = count - 2;
            }

            System.out.println();
        }
    }
}
```

OUTPUT

```
Input the number:  7
      1
     212
    32123
   4321234
  543212345
65432123456
7654321234567
65432123456
  543212345
   4321234
```

```
32123
 212
  1
```

C# SHARP

CODING EXERCISES

CODING FOR BEGINNERS

JJ TAM

C# SHARP CODING EXERCISES

Print hello and your name

in a separate line in C#

C# CODE

```
public class Exercise1
{
   public static void Main( )
   {
       System.Console.WriteLine("Hello");
       System.Console.WriteLine("JJ TAM!");
   }
}
```

Output:

```
Hello

JJ TAM
```

Print the result of the specified operations

C# CODE

```
public class Exercise4
{
    public static void Main()
    {
        System.Console.WriteLine(-1+4*6);
         //-1 + 24 = 23
        System.Console.WriteLine((35+5)%7);
         //40 % 7 = 5 (remainder of 40/7)
        System.Console.WriteLine(14+-4*6/11);
        //14 - (24/11)= 14 - 2 = 12
        System.Console.WriteLine(2+15/6*1-7%2);
         //2 + (15/6) - remainder of (7/2) = 2 + 2 - 1 = 4 - 1 = 3
    }
}
```

Output:

```
23


5
```

12

3

Program to swap two numbers

C# CODE

```
using System;
public class Exercise5
{
      public static void Main(string[] args)
       {
          int number1, number2, temp;
          Console.Write("\nInput the First Number : ");
          number1 = int.Parse(Console.ReadLine());
          Console.Write("\nInput the Second Number : ");
          number2 = int.Parse(Console.ReadLine());
          temp = number1;
          number1 = number2;
          number2 = temp;
          Console.Write("\nAfter Swapping : ");
          Console.Write("\nFirst Number : "+number1);
          Console.Write("\nSecond Number : "+number2);
          Console.Read();
      }
}
```

Output:

```
Input the First Number : 2

Input the Second Number : 5
```

```
After Swapping
:
First Number :
5
Second Number : 2
```

Print the output of multiplication

of three numbers

which will be entered by the user

C# CODE

```
using System;
public class Exercise6
{
  public static void Main()
  {
    int num1, num2, num3;

    Console.Write("Input the first number to multiply: ");
    num1 = Convert.ToInt32(Console.ReadLine());

    Console.Write("Input the second number to multiply: ");
    num2 = Convert.ToInt32(Console.ReadLine());

    Console.Write("Input the third number to multiply: ");
    num3 = Convert.ToInt32(Console.ReadLine());

    int result = num1 * num2 * num3;
    Console.WriteLine("Output: {0} x {1} x {2} = {3}",
                        num1, num2, num3, result);
  }
}
```

Output:

```
Input the first number to multiply:
2
Input the second number to multiply:
8
Input the third number to multiply:
5
Output: 2 x 8 x 5 = 80
```

Adding, subtracting, multiplying and dividing

of two numbers which

will be entered by the user

C# CODE

```
using System;
public class Exercise7
{
    public static void Main()
    {
        Console.Write("Enter a number: ");
        int num1= Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter another number: ");
        int num2= Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("{0} + {1} = {2}", num1, num2, num1+num2);
        Console.WriteLine("{0} - {1} = {2}", num1, num2, num1-num2);
        Console.WriteLine("{0} x {1} = {2}", num1, num2, num1*num2);
        Console.WriteLine("{0} / {1} = {2}", num1, num2, num1/num2);
        Console.WriteLine("{0} mod {1} = {2}", num1, num2,
num1%num2);
   }
}
```

Output:

```
Enter a number:
10
Enter another number:
2
10 + 2 =
12
10 - 2 =
8
10 x 2 =
20
10 / 2 =
5
10 mod 2 = 0
```

Print the average of four numbers

C# CODE

```
using System;
using System.IO;
public class Exercise9
{
  public static void Main()
  {
     double number1,number2,number3,number4;

     Console.Write("Enter the First number: ");
     number1 = Convert.ToDouble(Console.ReadLine());

     Console.Write("Enter the Second number: ");
     number2 = Convert.ToDouble(Console.ReadLine());

     Console.Write("Enter the third number: ");
     number3 = Convert.ToDouble(Console.ReadLine());

     Console.Write("Enter the fourth number: ");
     number4 = Convert.ToDouble(Console.ReadLine());

     double result = (number1 + number2 + number3 + number4) / 4;
     Console.WriteLine("The average of {0}, {1}, {2}, {3} is: {4} ",
  number1, number2, number3, number4, result);
   }
}
```

Output:

```
Enter the First number:
17
Enter the Second number:
17
Enter the third number:
517
Enter the four number:
51
The average of 17, 17, 517, 51 is: 150.5
```

Program to convert Temperature

from celsius degrees

to Kelvin and Fahrenheit

kelvin = celsius + 273

fahrenheit = celsius x 18 / 10 + 32

C# CODE

```
using System;
public class Exercise14
{
  public static void Main( )
   {
      Console.Write("Enter the amount of celsius: ");
      int celsius = Convert.ToInt32(Console.ReadLine());

      Console.WriteLine("Kelvin = {0}", celsius + 273);
      Console.WriteLine("Fahrenheit = {0}", celsius * 18 / 10 + 32);
   }
}
```

Output:

```
Enter the amount of celsius:
40
Kelvin =
313
Fahrenheit = 104
```

Compute the sum

of two given integers,

if two values are equal then return the triple of their sum

C# CODE

```
using System;
using System.Collections.Generic;

public class Exercise19 {
  static void Main(string[] args)
        {
            Console.WriteLine(SumTriple(2,2));
            Console.WriteLine(SumTriple(12,10));
            Console.WriteLine(SumTriple(-5,2));
        }
        public static int SumTriple(int a, int b)
        {
            return a == b ? (a + b)*3 : a + b;
        }
}
```

Output:

```
12
```

22

-3

Compute sum

of all the elements

of an array of integers

C# CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

public class Exercise47
{
   public static void Main()
      {
         int[] nums = {1, 2, 2, 3, 3, 4, 5, 6, 5, 7, 7, 7, 8, 8, 1};
         Console.WriteLine("\nArray1: [{0}]", string.Join(", ", nums));
         var sum = 0;
            for (var i = 0; i < nums.Length; i++)
            {
                sum += nums[i];
            }
           Console.WriteLine("Sum: "+ sum);
    }
}
```

Output:

```
Array1: [1, 2, 2, 3, 3, 4, 5, 6, 5, 7, 7, 7, 8, 8, 1]

Sum: 69
```

C# program to check

if an array contains an odd number

CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

public class Exercise53
{
   public static void Main()
      {
         int[] nums = {2, 4, 7, 8, 6};
         Console.WriteLine("\nOriginal array: [{0}]", string.Join(", ",
nums));
         Console.WriteLine("Check if an array contains an odd number?
"+even_odd(nums));
      }

    public static bool even_odd(int[] nums)
    {
         foreach (var n in nums)
            {
                if (n % 2 != 0)
                return true;
            }
        return false;

    }
}
```

OUTPUT

```
Original array: [2, 4, 7, 8, 6]

Check if an array contains an odd number? True
```

C# Sharp program to check two given integers

and return true if one of them is 30 or if their sum is 30

C# CODE

```
using System;
namespace exercises
{
     class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(test(30, 0));
            Console.WriteLine(test(25, 5));
            Console.WriteLine(test(20, 30));
            Console.WriteLine(test(20, 25));
            Console.ReadLine();
        }

      public static bool test(int x, int y)
        {
            return x == 30 || y == 30 || (x + y == 30);
        }
```

```
  }
}
```

Output:

```
True

True

True

False
```

C# Sharp program to check a given positive number

is a multiple of 3 or a multiple of 7

C# CODE

```
using System;
namespace exercises
{
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(test(3));
            Console.WriteLine(test(14));
            Console.WriteLine(test(12));
            Console.WriteLine(test(37));
            Console.ReadLine();
        }
        public static bool test(int n)
        {
            return n % 3 == 0 || n % 7 == 0;
        }
    }
}
```

Output:

True

True

True

False

check whether a given string starts with 'C#' or not

C# CODE

```
using System;
namespace exercises
{
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(test("C# Sharp"));
            Console.WriteLine(test("C#"));
            Console.WriteLine(test("C++"));
            Console.ReadLine();
        }
   public static bool test(string str)
        {
            return (str.Length < 3 && str.Equals("C#")) ||
(str.StartsWith("C#") && str[2] == ' ');
        }
    }
}
```

Output:

```
True
```

True

False

C# Sharp program to check the largest number

among three given integers

C# CODE

```
using System;

namespace exercises
{
  class Program
   {
       static void Main(string[] args)
       {
           Console.WriteLine(test(1,2,3));
           Console.WriteLine(test(1,3,2));
           Console.WriteLine(test(1,1,1));
           Console.WriteLine(test(1,2,2));
           Console.ReadLine();
       }

      public static int test(int x, int y, int z)
       {
           var max = Math.Max(x, Math.Max(y, z));
           return max;
       }
```

```
    }
}
```

Output:

```
3

3

1

2
```

Program to check which number is nearest to the value 100

among two given integers

Return 0 if the two numbers are equal.

C# CODE

```
using System;
namespace exercises
{
   class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(test(78, 95));
            Console.WriteLine(test(95, 95));
            Console.WriteLine(test(99, 70));
            Console.ReadLine();
        }
        public static int test(int x, int y)
        {
            const int n = 100;
            var val = Math.Abs(x - n);
            var val2 = Math.Abs(y - n);

            return val == val2 ? 0 : (val < val2 ? x : y);
        }
    }
}
```

Output:

```
95

0

99
```

Program to convert the last 3 characters

of a given string in upper case

If the length of the string has less than 3 then uppercase all the characters.

C# CODE

```
using System;
namespace exercises
{
   class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(test("Python"));
            Console.WriteLine(test("Javascript"));
            Console.WriteLine(test("js"));
            Console.WriteLine(test("PHP"));
            Console.ReadLine();
        }
     public static string test(string str)
```

```
        {
            return str.Length < 3 ? str.ToUpper() : str.Remove(str.Length -
3) + str.Substring(str.Length - 3).ToUpper();
        }
  }
}
```

Output:

```
PytHON

JavascrIPT

JS

PHP
```