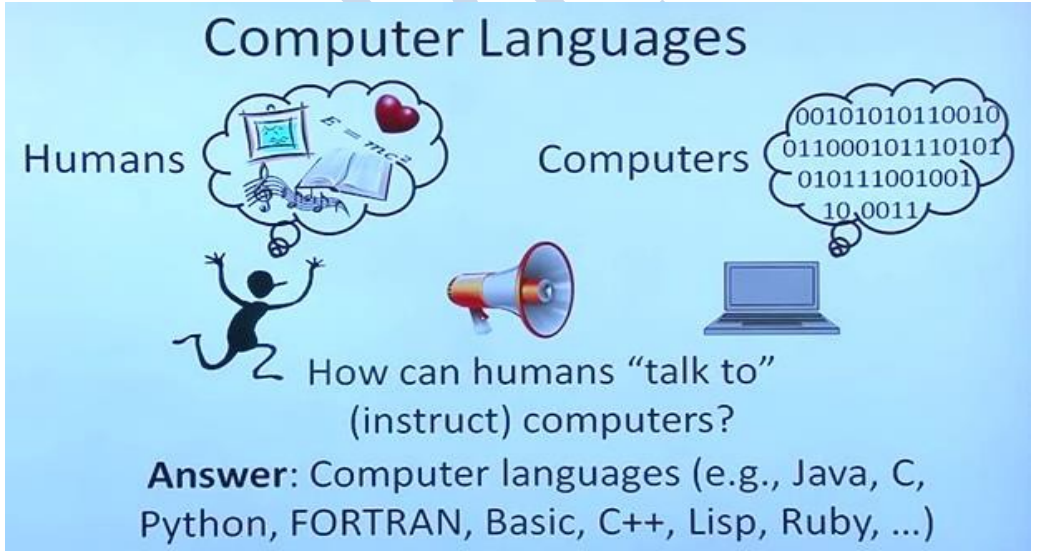
Unit II

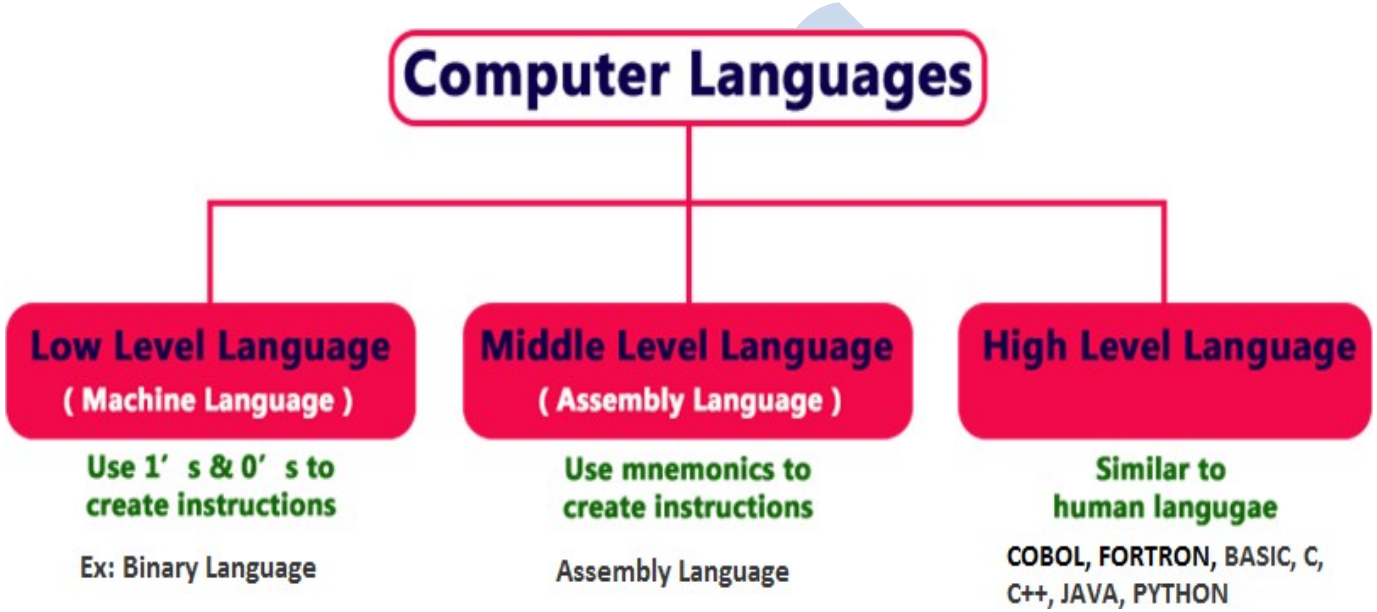
COMPUTER LANGUAGES

- Computer languages are the languages through which user can communicate with the computer by writing program instructions.



Computer Language

Types of Languages



Types of Languages

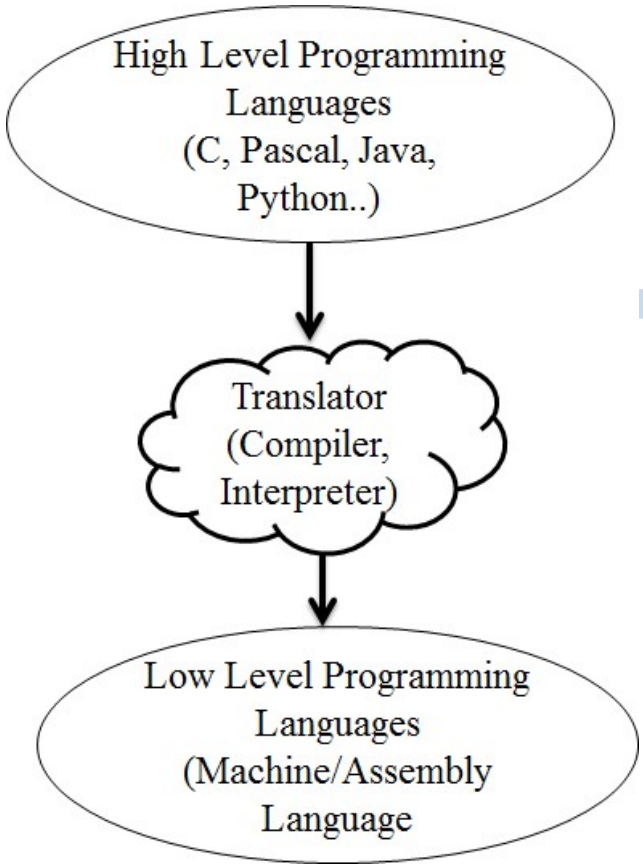
Why Translator?

- High-level programming language ➔ is written in English language
  - Humans can understand
  - But computers can understand only low level languages

- High level programming language ➔ Converted to ➔ Low level programming language.
  - Its done with the translator

APEC

Contd…


High Level Programming Languages (C, Pascal, Java, Python..)
Translator (Compiler, Interpreter)
Low Level Programming Languages (Machine/Assembly Language


PYTHON
PERL
C#
RUBY

Introduction to Python

- Python is an example of a high level Programming language.
  - Python language begins its implementation in December 1989 by Guido van Rossum
  - Python is an interpreted, object-oriented, high-level programming language with dynamic semantics
  - Python 3.0 - December 3, 2008
  - Python 3.6 - December 23, 2016

APC

Features of Python

- **Simple**
  - Python is a simple language.
  - Its like reading the English language
- **Easy to Learn**
  - Python is extremely easy to get started with.
  - Python has an extraordinarily simple syntax.
- **Free and Open Source**
  - Python is an example of a FLOSS (Freeand Open Source Software).
- **High-level Language**
  - When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

APEC

Features of Python – Contd…

- **Portable**
  - Due to its open-source nature, Python can be worked on any platforms. Python can work on Linux, Windows, Solaris etc.
- **Interpreted**
  - Programs written in C or C++ is converted into a language that is understood by the computer (binary code i.e. 0s and 1s).
  - But Python, on the other hand, does not need this translation. Python converts the source program into an intermediate form called bytecodes, which is used by the computer.

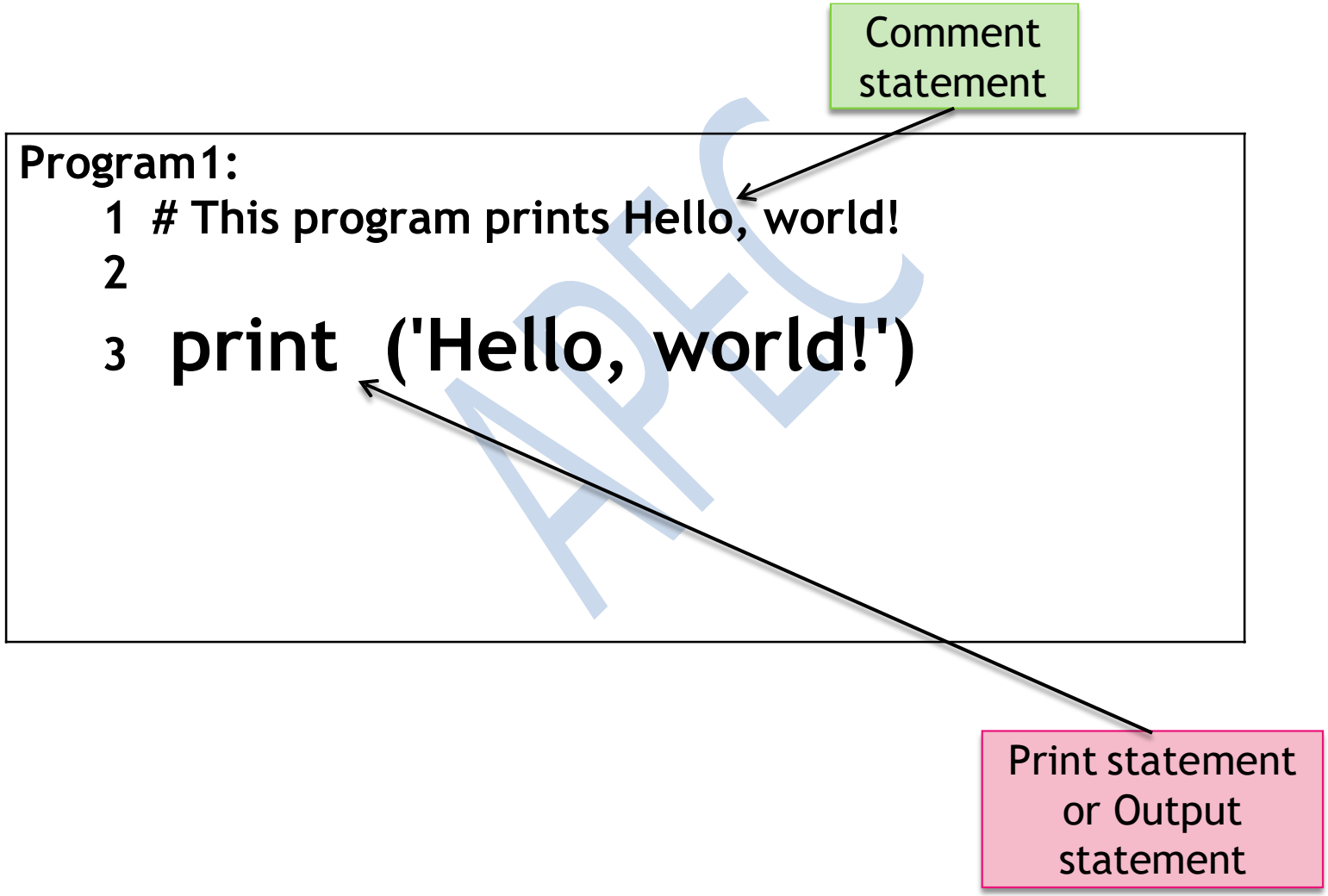
APEC

Features of Python – Contd…

- **Object Oriented**
  - Python supports procedure-oriented programming as well as object-oriented programming.
- **Embeded**
  - You can embed Python within your C/C++ programs
- **Extensive Libraries**
  - The Python Standard Library is huge indeed.
  - It helps in unit testing, databases, web browsers, CGI, ftp, email, cryptography, GUI (graphical user interfaces).

APEC

Simple Python Program



String Program to Welcome

```
name = input("Enter your name :")
greeting = "Good Morning"
print (greeting + name)
```

Output

```
Enter your name : CSE Students
Good Morning CSE Students
```

APEC

Python modes

- **Normal mode**
  - Here, the scripted and finished .py files are run in the Python interpreter. This mode is also called as script mode
- **Interactive mode**
  - It's the command line shell which gives immediate feedback for each statement

APEC

Python Input and Output Statements

- **PYTHON OUTPUT STATEMENT**
  - ✓ print() function is used to output .
  - ✓ *print* statement is used where zero or more expressions are passed, separated by commas.

- **PYTHON INPUT STATEMENT**

  Provides two built-in functions to read a line of text:

  input — Ex:num1 = input("Enter the Input")

  int(input) — Ex:num1 = int(input("Enter the Input"))

APEC

float(input) Ex:num1 = float(input("Enter the Input”))

APEC

Python program to get Employee details

```
# Get the name of the Employee
name = input("Enter the Employee name :")
# Get the age of the Employee
age = int(input("Enter the age of the Employee :"))
# Get the salary of the Employee
salary = float(input("Enter Emloyee salary :"))

print ("Employee Name : ", name)
print ("Employee Age : ", age)
print ("Employee salary : Rs. ", salary)
```

ADC

Output

```
Enter the Employee name : Rajesh
Enter the age of the Employee :45
Enter Emloyee salary : 67390
('Employee Name : ', ' Rajesh')
('Employee Age : ', 45)
('Employee salary : Rs. ', 67390.0)
```

APEC

Example

```
# Python program to print values

a = int(input("Enter number : "))
print ("Value = ", a)

b = float(input("Enter number : "))
print ("Value = ", b)

c = input("Enter string :")
print ("String = ", c)
# Calculations is done

# Calculations is done
print ("Calculations on the Input ")
print ("Value = ", a * 2)
```

Output

```
print ("Value = ", b * 2)
print ("String = ", c * 2)
```

APEC

Output

```
Enter number : 78 ('Value = ', 78)
Enter number : 78 ('Value = ', 78.0)
Enter string : Dhaya ('String = ', ' Dhaya')
Calculations on the Input ('Value = ', 156)
('Value = ', 156.0)
('String = ', ' Dhaya Dhaya')
```

APEC

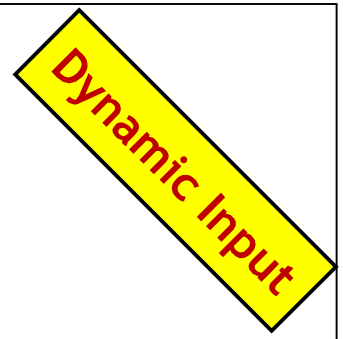
Static & Dynamic Inputs

- **Dynamic Inputs**
  - Inputs given at program execution time or run time
- **Static Inputs**
  - Inputs given in the program itself directly

APEC

Program: Addition of 2 numbers

```
a = int(input("Enter First number :"))
b = int(input("Enter Second number :"))
c = a + b
print ("Addition of two numbers :", c)
```



Output

```
Enter First number :78
Enter Second number :23
('Addition of two numbers :', 101)
```

APEC

Program: Addition of 2 numbers

Static Input

```
a = 45
b = 20
c = a + b
print ("Addition of two numbers :", c)
```

Output

```
('Addition of two numbers :', 65)
```

APEC

Comment statement

```
# Program to add two numbers
a = 56
b = 1
c = a + b
print ("Addition of two numbers :", c)
```

Comment statement in Python

Started with # symbol

- Comment statements will not executed
- They give the explanation of the program

APEC

Python Comments

- Single line comment:

```
#This is a comment statement
print('Hello')
```

  - Multiple Line Comment:

```
# This is a long comment
# and it extends
# to multiple lines
```

  - Another way to represent multiple comment is to use triple quotes.

```
“ “ “This is also a
perfect example of
multi-line comments” ” ”
```

APEC

Python Variables

Define Variable.

- Variables are reserved memory location that store values such as numbers , text , etc.

- In python declaration of variable happens automatically.

```
addition = 900          # An integer assignment
amount  = 1000.0        # A floating point assignment
name    = "XXX"         # A string assignment
```

- Multiple objects assigned to multiple variables.

```
addition, amount, name = 900,1000.0,"john"
```

APEC

Python Variables – contd…

- Single value can be assigned to multiple variables.

```
mark1 = mark2 = mark3 = 50
```

- Python swap values in a single line and this applies to all objects in python.

```
var1, var2 = var2, var1
```

APEC

Example

```
# Program to display values
count = 500
print ("Count = ", count)
```

Output

```
('Count = ', 500)
```

APEC

Example

```
# Program to display values
age = 25
Department = "Computer Science & Engineering"
College = "Adhiparasakthi Engineering College“

print ("My Age : ", age)
print ("My Department :", Department)
print ("My College :", College)
```

Output

Output

```
('My Age : ', 25)
('My Department :', 'Computer Science & Engineering')
```

APEC

('My College :', 'Adhiparasakthi Engineering College')

APEC

Example

```
# Program to assign values
a,b = 56,20
print ("A value :", a)
print (“B value :", b)
```

Output

```
('A value :', 56)
('B Value :', 20)
```

APEC

Example

```
# Program to add two numbers
a,b = 34,89
print ("Addition of two numbers = ", a+b)

a,b = 56,20
c = a + b
print ("Addition of two numbers :", c)
```

Output

```
('Addition of two numbers = ', 123)
('Addition of two numbers :', 76)
```

APEC

Example

```
# Program to swap (exchange) two numbers
a,b = 56,20
print ("Before swapping", a, b)
a,b = b,a
print ("After swapping", a, b)
```

Output

```
('Before swapping', 56, 20)
('After swapping', 20, 56)
```

APEC

Example

```
# Program to display values
a = 25
b = 45
print ("First time values ", a , b)

a = 90
b = 67
print ("Second time values", a , b)
```

Output

```
('First time values ', 25, 45)
('Second time values', 90, 67)
```

APEC

Python Identifiers or Python Variables

- ***Rules for defining Identifiers:***
  - Identifiers can be a combination of lowercase letters (a to z) or uppercase letters (A to Z) or digits (0 to 9) or an underscore (_).
  - identifiers cannot start with the digit.
  - Keywords cannot be used as identifiers and identifiers can be any length.
  - No special symbols like !, @, #, $, % etc. are used.

- ***Valid Identifiers:*** sum total average _ab_ add_1 x1
- ***Invalid Identifiers:*** 1x char x+y

APEC

Note:

- Python is case sensitive.
- Variable and variable are not the same.

APEC

Python Keywords

The list of all the keywords are given below:

and del from not while as elif global or with

assert else if pass yield break except import print class

exec raise in continue is finally return def for

lambda try True False None.

Note:

All the keywords except True, False and None are in

APEC

lowercase and they must be written as it is.

APEC

Python Statement

- End of a statement is marked by a newline character.
-
- Example:

```
statement =  x**2 + x**3 + x**4 + x**5 + x**6 + x**7 + x**8
```

- Statements can be extended over multiple lines using '\'.

```
statement = x + x**2 + x**3 \
          + x**4 + x**5 \
          + x**6 + x**7 \
                 + x**8
```

APEC

Python Statement (Cont..)

- Statements can be split into multiple lines using **‘()’,’{}’, and ‘[]’.**

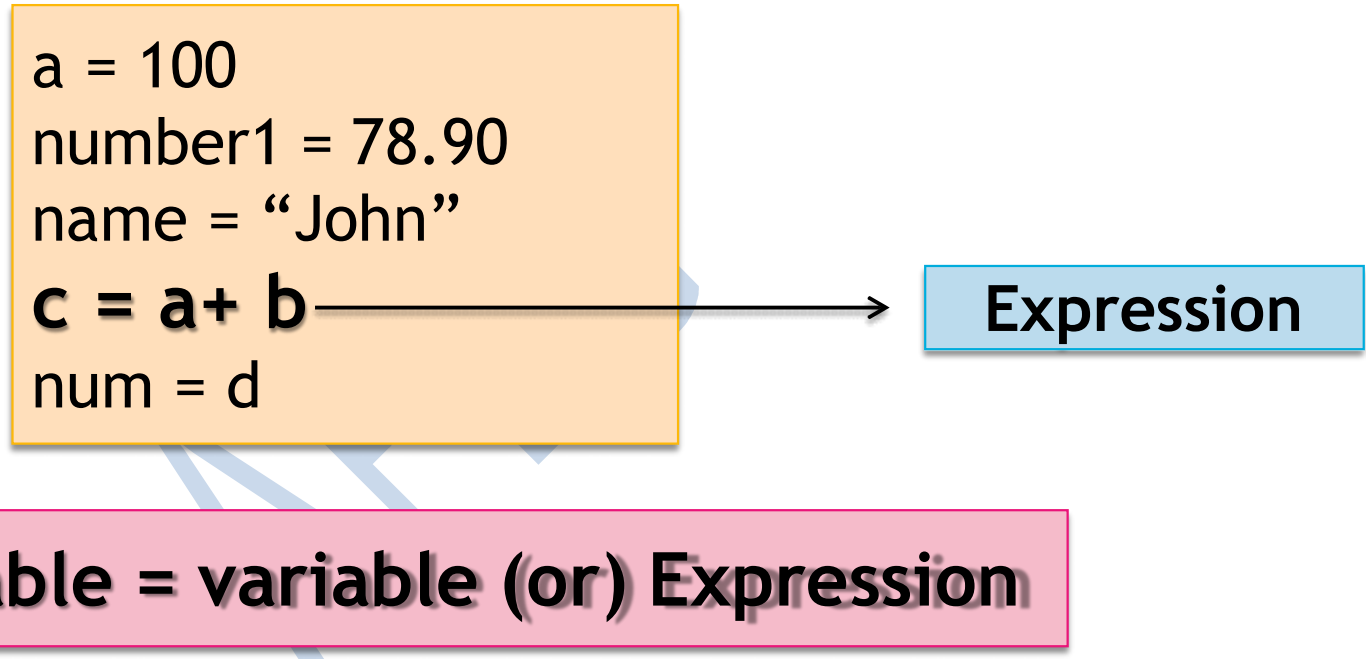
```
statement = (x + x**2/2 + x**3/3
             + x**4/4 + x**5/5
             + x**6/6 + x**7/7
             + x**8/8)
```

- Multiple statements can be put in same line using semicolon.

```
a = 1; b = 2; c = 3
```

APEC

Assignment Statement



- Assigns the value of the variable or the expression

Example

```
# Program to print values
a = 100
number1= 78.90
name = "John"
a = 33
b = 8
c = a + b
print ("Value of a :", a)
print ("Value of b :", b)
print ("Value of c :", c)
print ("Value of number1 :", number1)
print ("Value of name :", name)

name = c
print ("Value of name :", name)
```

Variable name is initially initialized with “John”, but now changed

Output

```
('Value of a :', 33)
('Value of b :', 8)
('Value of c :', 41)
('Value of number1 :', 78.9)
('Value of name :', 'John')
```

APEC

('Value of name :', 41)

APEC

Program to calculate sum of 3 numbers

```
# Python program to calculate the sum of three numbers
a = int(input("Enter 1st number :"))
b = int(input("Enter 2nd number :"))
c = int(input("Enter 3rd number :"))
sum = a + b + c
print ("Addition of 3 numbers :", sum)
```

Output

```
Enter 1st number :8
Enter 2nd number :56
Enter 3rd number :34
```

APEC

('Addition of 3 numbers :', 98)

APEC

Simple Interest and Compound Interest

- $Simple\ Interest = (p * n * r)/100$
- $Compound\ Interest = p * [\left(1 + \frac{\square}{100}\right)^{\square}] - 1$
  - Principal amount p = 1000 rupees
  - Number of years n = 5 years
  - Rate of interest r = 10%
- $Simple\ interest = \frac{(1000*5*10)}{100} = 500$ □□□□□□
  - He has to return (Rs. 1000 + Rs. 500 = Rs. 1500/-)
- $Compound\ Interest = 1000 * [\left(1 + \frac{10}{100}\right)^{5}] - 1 = 610.51\ rupees$

APFC

- He has to return (Rs. 1000 + Rs. 610.51 = Rs. 1610.51/-)

APEC

Program to calculate Simple Interest

```
# Python program to calculate Simple Interest
# Simple Interest = (Principal * number of years * Rate of Interest)/100
principal_amount = float(input("Enter Principal Amount :"))
rate_of_interest = int(input("Enter Rate of Interest :"))
Number_of_years = int(input("Enter  number of years :"))
simple_interest = (principal_amount * rate_of_interest * Number_of_years)/100
print ("Simple Interest is Rs. ", simple_interest)
```

Output

Output

```
Enter Principal Amount : 10000
Enter Rate of Interest : 9
```

```
Enter  number of years : 3
('Simple Interest is Rs. ', 2700.0)
```

APEC

Program to find Total & Average

```
# Python program to calculate Total marks obtained in 5 subjects
# Average of the student
name = input("Enter name of the student :")
mark1 = int(input("Enter Mark 1: "))
mark2 = int(input("Enter Mark 2: "))
mark3 = int(input("Enter Mark 3: "))
mark4 = int(input("Enter Mark 4: "))
mark5 = int(input("Enter Mark 5: "))

total = mark1+mark2+mark3+mark4+mark5
average = total/5

print ("Name of the Student : ", name)
print ("Total marks obtained : ", total)
```

```
print ("Average marks obtained : ", average)
```

APEC

Program – Contd…

Output

```
Enter name of the student : Vasu
Enter Mark 1: 89
Enter Mark 2: 90
Enter Mark 3: 78
Enter Mark 4: 96
Enter Mark 5: 89
('Name of the Student : ', ' Vasu')
('Total marks obtained : ', 442)
('Average marks obtained : ', 88)
```

APEC

Program to calculate Area and Circumference of the Triangle

```
# Calculate Area and Circumference of the Triangle
# Area = pi * r * r
# Circumference = 2 * pi * r

r = int(input("Enter radius: "))
Area = pi * r * r
Circumference = 2 * pi * r

print ("Area of the Circle = ", Area)
print ("Circumference of the Circle = ", Circumference)
```

APEC

Contd...

Output

```
Enter radius: 90

Traceback (most recent call last):
  File "C:/Python27/p1.py", line 6, in <module>
    Area = pi * r * r
NameError: name 'pi' is not defined
```

APEC

Program to calculate Area and Circumference of the Triangle

```
# Calculate Area and Circumference of the Triangle
# Area = pi * r * r
# Circumference = 2 * pi * r

r = int(input("Enter radius: "))
Area = 3.14 * r * r
Circumference = 2 * 3.14 * r

print ("Area of the Circle = ", Area)
print ("Circumference of the Circle = ", Circumference)
```

APEC

Output

```
Enter radius: 9
('Area of the Circle = ', 254.34)
('Circumference of the Circle = ', 56.52)
```

APEC

Python program to convert Celsius to Fahrenheit and vice-versa

- Fahrenheit ➔ Celsius
  - $Celcius = (Fahrenheit - 32) * 5/9$
- Celsius ➔ Fahrenheit
  - $Fahrenheit = (Celcius * 9/5) + 32$

APEC

Program to convert temperature in celsius to fahrenheit

```
# Python Program to convert temperature in celsius to fahrenheit
celsius = 37.5
# calculate fahrenheit
fahrenheit = (celsius * 9/5) + 32
print("Fahrenheit = ", fahrenheit)
```

Output

```
('Fahrenheit = ', 99.5)
```

APEC

Python Datatypes

Python Datatypes

Python has five standard data types:

- Numbers
- String
- List
- Tuple
- Set
- Dictionary

APEC

Numbers

- Python has three number types:
- Integer numbers
  - Integers can be of any length and is limited to the memory size.
  - A number prefixed by a 0 (zero) is interpreted as an octal number. Hexadecimal numbers are prefixed either by **"0x"** or **"0X"**.
- Floating point numbers
  - Floating point numbers is restricted to 10 decimal places.
- Complex numbers.
  - Complex numbers are of the form, **x + yj**, where x is the

APEC

real part and y is the imaginary part.

APEC

Program to print decimal, octal and hexadecimal number

```
# Print Decimal, Octal, Hexadecimal number
a = 10
print (a)

a = 010
print (a)

a = 0x10
print (a)
```

Output

```
10
8
16
```

APEC

Print the decimal value of the given number
010 ➔ octal number, corresponding decimal value = 8

0x10 ➔ Hexadecimal number, corresponding decimal value = 16

APEC

Floating point numbers

- Floating point numbers is restricted to 10 decimal places (depends on python versions).

```
# Program to print floating number
a = 10872899283298239823023 0
print (a)
a = 10.87289928329823982302 30
print (a)
```

Output

```
10872899283298239823023 0
10.8728992833
```



Last value is rounded

APEC

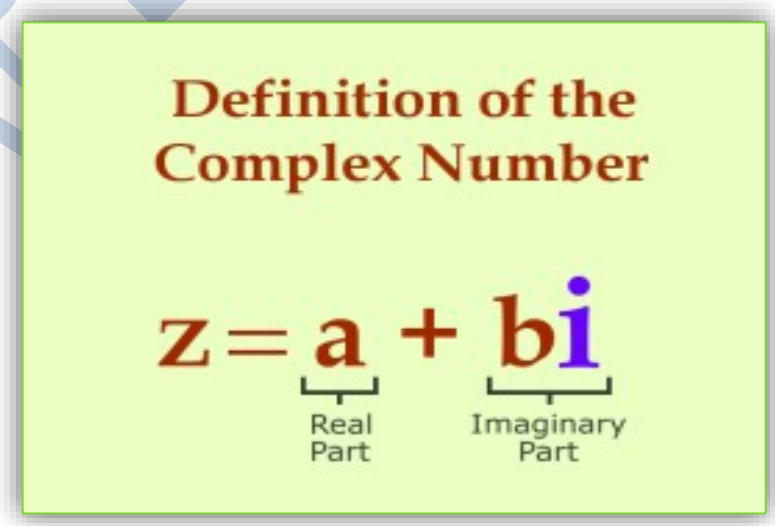
Complex numbers

- Complex numbers are of the form, x + yj, where x is the real part and y is the imaginary part.

```
# Program to print complex number
a = 78+90j
print (a)
```

Output

```
(78+90j)
```



Contd…

```
#Python program to add two complex numbers
a = input("Enter complex number 1 :")
b = input("Enter complex number 2 :")
c = a + b
print ("Addition of two complex numbers = ", c)
```

Output

```
Enter complex number 1 : 2+6j
Enter complex number 2 : 3+8j
('Addition of two complex numbers = ', (5+14j))
```

APEC

Program to convert number types

```
# Python program to convert number types
a = float(15)
print ("15 in float = ", a)
a = int(34.89)
print ("34.89 in int = ", a)
a = complex(56)
print ("56 in complex = ", a)
a = complex(-7.78)
print ("56 in complex = ", a)
```

Output

Output

('15 in float = ', 15.( )

APEC

```
('34.89 in int = ', 34)
('56 in complex = ', (56+0j))
('56 in complex = ', (-7.78+0j))
```

APEC

Program to convert number types

```
# Python program to convert number types
print ("154 in Hexadecimal = ", hex(154))
print ("15 in binary = ", bin(15))
print ("56 in Octal = ", oct(56))
```

Output

Output

```
('154 in Hexadecimal = ', '0x9a')
('15 in binary = ', '0b1111')
('56 in Octal = ', '070')
```

APEC

Type Promotion (or) Type coercion

```
# Python program to convert number types
a = 20
b = 67.95
c = a + b
print c
```

Integer is promoted to float

```
a = 20
b = 6 + 7j
c = a + b
print c
```

Integer is promoted to complex

```
a = 20
b = "Hi"
c = a + b
print c
```

Output

Output

87.95

(26+7j)

TypeError: unsupported operand type(s) for +: 'int' and 'str'

APEC

APEC

Delete statement

- A variable is deleted using del statement
- **Syntax:** del variable name

```
# Program to delete a number
a = 67
print ("Value of a = ", a)

del a
print ("Value of a = ", a)
```

Output

```
('Value of a =', 67)

Traceback (most recent call last):
  File "C:/Python27/p2.py", line 5, in
<module>
    print ("Value of a =", a)
```

APEC

NameError: name 'a' is not defined

APEC

Special Functions in Python

- **type()** function ➔ Used to check the class a variable
- **isinstance()** function ➔ Used to check if an object belongs to a particular class

APC

Special Functions in Python


```
# Program to use special functions
print (type(45))
print (type(89.67))
print (type("CSE"))
print (type(3+4j))

print (isinstance(3+4j,complex))
```

Output

```
<type 'int'>
<type 'float'>
<type 'str'>
<type 'complex'>
True
```

APEC

Example

```
#python program
name = input("Enter name :")
age = int(input("Enter age :"))
salary = float(input("Enter salary :"))

print ("Name is of ", type(name))
print ("Age is of ", type(age))
print ("Salary is of", type(salary))
```

Output

```
Enter name :Raji
Enter age :67
Enter salary :6878.90
('Name is of ', <type 'str'>)
('Age is of ', <type 'int'>)
```

ARC

('Salary is of', <type 'float'>)

APEC

Strings

- String is a sequence of characters, numbers, and special characters enclosed within single-quotes(‘ ‘) or double-quote (“ ”).

```
str = "Python Programming"
print (str)

str = input("Enter the String :")
print (str)
```

Output

Output

Python Programming

Enter the String : My name

My nam e is Raj

APEC

is Raj

APEC

More on strings

```
str1 = ' Inside single Quote '
str2 = " Inside double Quote "
str3 = """ Inside triple Quote
           multiline
           statement   """

print (str1)
print (str2)
print (str3)
```

Output

Output

```
Inside single Quote
 Inside double Quote
 Inside triple Quote
           multiline
           statement
```

APEC

APEC

More on strings

```
str = "My class room"
print (str)
print (str[0])        # Indexing - Prints first character of the string
print (str[3:7])      # slicing - Prints characters starting from 3rd to 7th
print (str[3:])
print (str[:])
print (str[:10])


str = "Python"
print (str + "program") # concatenati
```

Output

```
My class room
M
clas
class room
My class room
My class r
```

APEC

Pythonprogram

APEC

Example

```
#python program to explain string addition
str1 = "II CSE"
str2 = "Students"
print ("string Addition = ", str1 + str2)

str1 = 100
str2 = "Students"
print ("string Addition = ", str1 + str2)
```

Output

Output

```
('string Addition = ', 'II CSEStudents')

Traceback (most recent call last):
  File "C:/Python27/p1.py", line 8, in <module>
    print ("string Addition = ", str1 + str2)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

APEC

>>>

APEC

Example

```
#python program to explain string addition
str1 = "II CSE"
str2 = "Students"
print ("string Addition = ", str1 + str2)

str1 = “100”
str2 = "Students"
print ("string Addition = ", str1 + str2)
```

APEC

Output

```
('string Addition = ', 'II CSEStudents')
```

('string Addition = ', '100Students')

APEC

Example

```
#python program to explain string addition
str1 = "II CSE"
str2 = "Students"
print (str1[3:6]+" year " + str2)
```

APEC

Output

CSE year Students

Lists

- List is an **ordered sequence of** elements of mixed data type.
- List is created by placing all the elements in a square bracket [ ], separated by commas.

APC

Example

```
list1 = ["python", "php"]
list2 = ["Big Data", "Hadoop", "R Programming"]
print (list1 + list2)
print (list1 * 2)
```

Output

```
['python', 'php', 'Big Data', 'Hadoop', 'R Programming']
['python', 'php', 'python', 'php']
```

APEC

Lists - Contd...


```
list1= [  ]                          # Empty list
list2= [ 67,90,12,88]                # List with integer elements
list3 = ['python', 6732, 45.23, 'jack', 100.2]

# Prints first element of the list3
print ("First element of List 3 =", list3[0])

# Prints elements starting from 2nd till 3rd
print ("Sliced List 3 = ", list3[1:3])

print ("Repeated List 3 =", list3 * 2)      # Prints list3 two times
```

APEC

```
print ("Concatenated List 2 & List 3 =", list2 + list3)
```

APEC

Contd…

Output

First element of List 3 = python

Sliced List 3 = [6732, 45.23]

Repeated List 3 = ['python', 6732, 45.23, 'jack', 100.2, 'python', 6732, 45.23, 'jack', 100.2]

Concatenated List 2 & List 3 = [67, 90, 88, 'python', 6732, 45.23, 'jack', 100.2]

ADEC

Tuples

A tuple consists of a **ordered**, sequence of elements separated by commas. Tuples are enclosed in parentheses ( ) and cannot be updated.

```
#Simple Tuple Example
tup3 = ('hello', 23, 34.90, 'world')
print ("Length of the Tuple = ", len(tup3))
print ("Maximum value in the Tuple= ", max(tup3))
print ("Minimum value in the Tuple= ", min(tup3))
del tuple3
```

Output

```
Length of the Tuple =  4
Maximum value in the Tuple=  world
Minimum value in the Tuple=  23
```

APEC

```
del tuple3
NameError: name 'tuple3' is not defined
```

APEC

Difference between Python Lists and Tuples

LIST (Data Type in Python)	TUPLE (Data type in Python)
Lists are enclosed in square brackets [ ]	Tuples are enclosed in square brackets ( )
Elements in the list can be changed	Elements in the tuples cannot be changed
listname = [89,90.9,"list"] print (listname) listname[2] = 3784 print (listname) **Output:** [89, 90.9, 'list'] [89, 90.9, 3784]	tuplename = (89,90.9,"list") print (tuplename) tuplename[2] = 3784 print (tuplename) **Output:** (89, 90.9, 'list') tuplename[2] = 3784 TypeError: 'tuple' object does not support item assignment

APC

```
# Python program to explain immutability in Tuples
tuple2= (78,90,23,787)
tuple3 = ('hello', 673, 34.90, 'world')
list3 = ['hello', 673, 34.90, 'world']       # check list elements brackets []

print ("Tuple 2 = ", tuple2 )
print ("First element of Tuple 2 =", tuple2[0])
print ("Concatenated Tuple 2 & Tuple 3 =", tuple2 + tuple3)
print ("Tuple 3 = ", tuple3)
Print ("List 3 = ", list3)

list3 = [1000,89,'hi']
print ("Changed List 3 = ", list3)
tuple3 = (1000,89,'hi')
print ("Changed Tuple 3 = ", tuple3)

list3[2] = 5555
print ("Changed one element of List 3 = ", list3)
tuple3[2] = 5555
```

APEC

```
tuple3[2] = 5555
print ("Changed one element of Tuple 3 = ", tuple3)
```

APEC

Output

```
Tuple 2 =  (78, 90, 23, 787)
First element of Tuple 2 = 78
Concatenated Tuple 2 & Tuple 3 = (78, 90, 23, 787,
'hello', 673, 34.9, 'world')
Tuple 3 =  ('hello', 673, 34.9, 'world')
List 3 =  ['hello', 673, 34.9, 'world']
Changed List 3 =  [1000, 89, 'hi']
Changed Tuple 3 =  (1000, 89, 'hi')
Changed one element of List 3  =  [1000, 89, 5555]
    tuple3[2] = 5555
TypeError: 'tuple' object does not support item
assignment
```

Sets

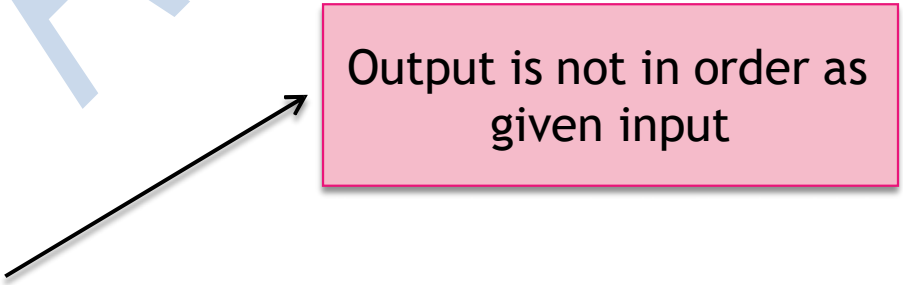
- Set is a **unordered c**ollection of unique and immutable objects.
- A set is created by placing all the elements inside curly braces {}, separated by comma.

Example:

```
x = set("python")
print x
```

Output:

```
set(['h', 'o', 'n', 'p', 't', 'y'])
```



APEC

Sets - Contd..

- Set do not accept duplicate elements.

Example

```
cities = set(("Madurai","Chennai","Erode", "salem", "Madurai","erode"))
print (cities)
```

Output

```
set(['salem', 'erode', 'Madurai', 'Chennai', 'Erode'])
```

Example

```
cities = set("Madurai","Chennai","Erode", "salem", "Madurai","erode")
print (cities)
```

Bracket missing

Output

```
Traceback (most recent call last):
  File "C:/Python27/p2.py", line 1, in <module>
    cities = set("Madurai","Chennai","Erode", "salem", "Madurai","erode")
```

APEC

TypeError: set expected at most 1 arguments, got 6

APEC

Sets - Contd..

- **add() and update() functions:** used to add new entries .
- **pop() or remove() or discard() function**: to remove entries.

APEC

Example


```
my_set={1,3}
print(my_set)
my_set.add(2)                    # add an element
print(my_set)
my_set.update([2,3,4])           # add multiple elements
print(my_set)
my_set.update([4,5], {1,6,8})    # add list and set
print(my_set)
#print my_set[0]   # TypeError: 'set' does not support indexing
my_set.discard(4)                # discard an element in a set
print(my_set)
my_set.remove(6)                 # remove an element
print(my_set)
my_set.pop()                     # pop 1st element from a set
print(my_set)
```

APEC

Discard & Remove

- If the element is not available
  - Remove raise a exception error
  - Discard does not raise any error

Program

```
a = {1,2,3,4,5,6,7,8,9}
a.discard(3)
print (a)                    # Output = set([1, 2, 4, 5, 6, 7, 8, 9])
a.discard(3)
print (a)                    # Output = set([1, 2, 4, 5, 6, 7, 8, 9])
a.remove(3)

Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    a.remove(3)
KeyError: 3
```

APEC

Output

```
set([1, 3])
set([1, 2, 3])
set([1, 2, 3, 4])
set([1, 2, 3, 4, 5, 6, 8])
set([1, 2, 3, 5, 6, 8])
set([1, 2, 3, 5, 8])
set([2, 3, 5, 8])
```

APEC

Example

print (set(a))

```
a = (1,2,3)
print (a)
print (list(a))
print (tuple(a))
print (set(a))
a="python"
print (a)
print (list(a))
print (tuple(a))
```

Output

APEC

set(['h', 'o', 'n', 'p', 't', 'y'])

Output

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

python
['p', 'y', 't', 'h',
'o', 'n']
('p', 'y', 't', 'h',
'o', 'n')

APEC

Example

```
a = (67,90.78,"Puppy",67,90.78,"puppy")
print (a)
print (len(a))
print (max(a))
print (min(a))
print (list(a*2))
print (tuple(a*2))
print (set(a*2))

a="python"
print (a)
print (list(a*2))
print (tuple(a*2))
print (set(a*2))
```

Output

```
(67, 90.78, 'Puppy', 67, 90.78, 'puppy')
6
puppy
67
[67, 90.78, 'Puppy', 67, 90.78, 'puppy', 67, 90.78, 'Puppy', 67, 90.78, 'puppy']
(67, 90.78, 'Puppy', 67, 90.78, 'puppy', 67, 90.78, 'Puppy', 67, 90.78, 'puppy')
set([67, 90.78, 'puppy', 'Puppy'])
python
['p', 'y', 't', 'h', 'o', 'n', 'p', 'y', 't', 'h', 'o', 'n']
('p', 'y', 't', 'h', 'o', 'n', 'p', 'y', 't', 'h', 'o', 'n') set(['h',
```

APEC

'o', 'n', 'p', 't',
'y'])

APEC

Dictionary

- Python dictionary is a container of the unordered set of elements like lists.The elements are surrounded by curly braces { }.
- Each element in the dictionary is a comma separated list of Key-Value pairs

APC

```
dict = {'Name': 'John', 'SSN': 4568, 'Designation': 'Manager'}

print "dict['Name']: ", dict['Name']

dict['Designation'] = 'Senior Manager' # update existing entry
dict['Location'] = 'New Delhi'  # Add new entry
```

APEC

Example

```
dict = {'Name': 'John', 'SSN': 4568, 'Designation': 'Manager'}
print ("dict['Name']: ", dict['Name'])
print ("dict['SSN']: ", dict['SSN'])
print ("dict['Designation']: ", dict['Designation'])

dict['Designation'] = 'Senior Manager‘ # update existing entry
dict['Location'] = 'New Delhi'  # Add new entry

print ("After Updating")
print ("dict['Designation']: ", dict['Designation'])
print ("dict['Location']: ", dict['Location'])
print ("Length of the Dictionary =", len(dict))
```

APEC

Output

dict['Name']: John
dict['SSN']: 4568
dict['Designation']: Manager
After Updating
dict['Designation']: Senior Manager
dict['Location']: New Delhi
Length of the Dictionary = 4

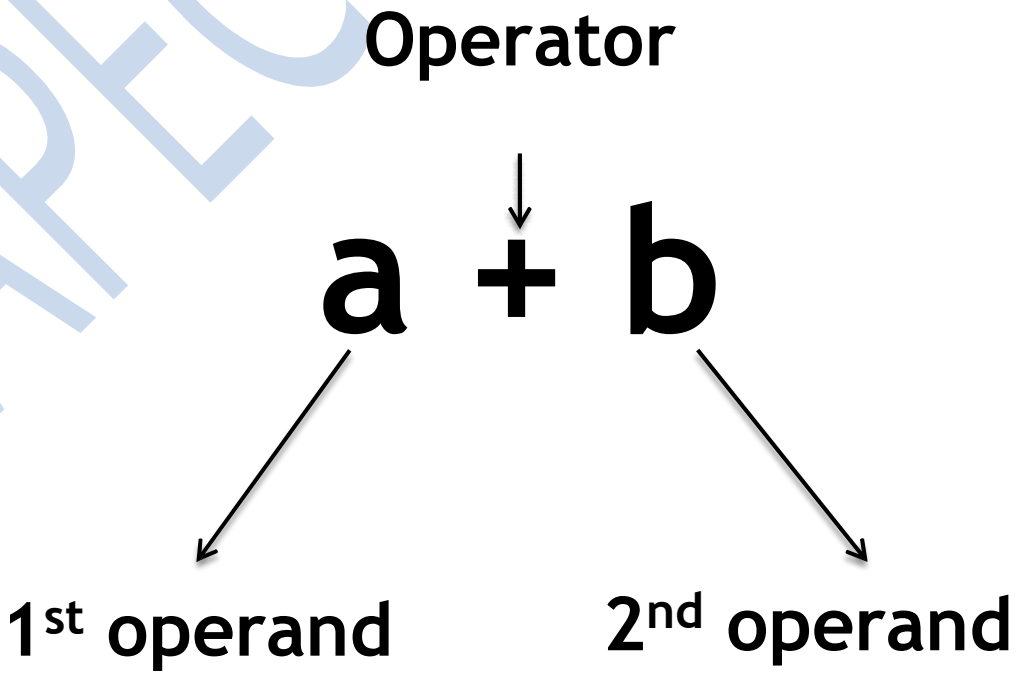
APEC

Python Operators

Python Operators

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Membership Operators
- Identity Operators



APEC

Arithmetic operators

- Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

Operator	Name	Expression	Example	Output
+	Addition	x+y	15 + 4	19
-	Subtraction	x-y	15 - 4	11
*	Multiplication	x*y	15 * 4	60
/	Division	x/y	15 / 4	3.75
%	Modulus	x%y	15 % 4	3
**	Exponentiation	x**y	15 ** 4	50625
//	Floor Division	x//y	15 // 4 15 // -4	3 - 4

APEC

Comparison operators

- Comparison operators are also called Relational operators. They are used to compare values and return True or False condition.

Operator	Name	Expression	Example	Output
>	Greater than	x>y	54 > 18	True
<	Lesser than	x<y	54 < 18	False
==	Equal to	x==y	54 == 18	False
!=	Not Equal to	x!=y	54 != 18	True
>=	Greater than or equal to	x>=y	54 >= 18	True
<=	Lesser than or equal to	x<=y	54 <= 18	False

AEC

Logical operators

- Logical operators used are AND, OR,NOT.

Operator	Name	Expression	Explanation
and	Logical and	x and y	True if both the operands are true
or	Logical or	x or y	True if either of the operands is true
not	Logical not	x not y	True if operand is false

Example

```
x = True
y = True
print (x and y)
print (x or y)
print (not x)

x = True
y = False
print (x and y)
print (x or y)
print (not x)
```

Output

```
True
True
False

False
True
False
```

APEC

Bitwise Operators

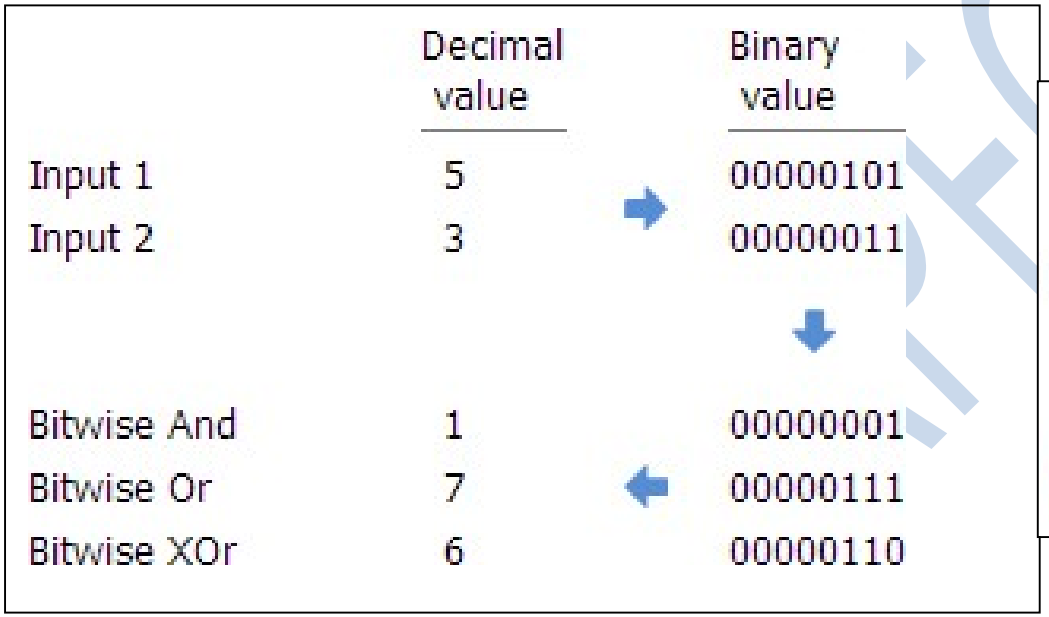
A	B	A | B	A & B	A^ B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

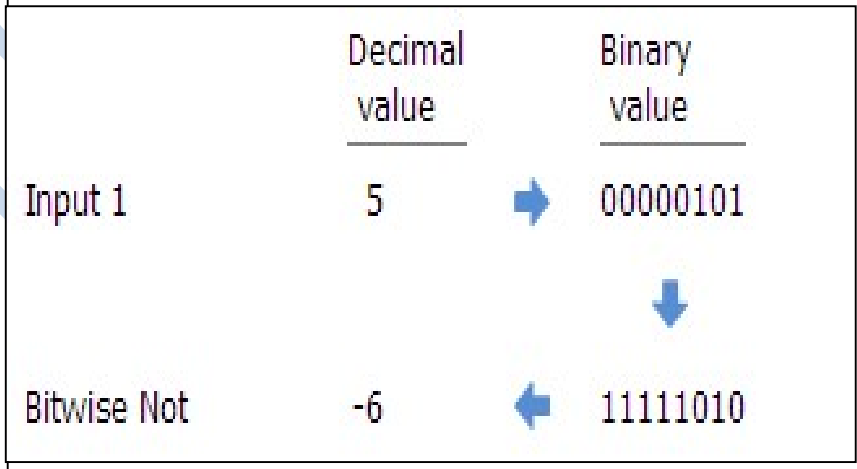
- **Bitwise AND**
  - Both inputs = 1, the output = 1.
  - If one or both bits are 0, the output = 0 for that bit
- **Bitwise OR**
  - one or both inputs = 1, the output is 1.
  - If both bits are 0, the output = 0 for the bit.
- **For Bitwise XOR**
  - one input is 1 and the other 0, the output = 1.

- If both bits are 1 or both are 0, the output is 0 for the bit.

APEC

Bitwise And, Or and XOr





Bitwise Operators

- Bitwise operation manipulates on bits.
- Numbers represented as bits(0's and1's).

Operator	Name	Example	Explanation
&	Bitwise AND	X & y	Both operands are true, then the result is true
|	Bitwise OR	x | y	If either of the operands are true, then the result is true
~	Bitwise NOT	~x	It complements the bits
^	Bitwise XOR	x ^ y	Result is true, when either of the operands are true, but not both
>>	Bitwise right shift	x>> 2	Operands are shifted right by the number of times specified
<<	Bitwise left shift	x<< 2	Operands are shifted left by the number of times specified

Example – Bitwise Operators

```
x = 60                          # In bitwise  60 = 0011 1100
y = 13                          # In bitwise, 13 = 0000 1101
z = 2
print('x & y  is',x&y)     #  0011 1100 & 0000 1101 = 0000 1100 = 12
print('x | y  is',x|y)     # 0011 1100 | 0000 1101  = 0011 1101 = 61
print('~x is',~x           # ~ 0011 1100 = -61 in 2's complement
print('x ^ y is',x^y)       # 0011 1100 ^ 0000 1101 = 0011 0001 = 49
print('x >> y is',x>>y )        # 0011 1100 shifted right 2 times
print('x << y is',x<<y)         # 0011 1100 shifted left 2 times
```

```
('x & y  is', 12)
('x | y  is', 61)
('~x is', -61)
```

```
('x ^ y is', 49)
('x >> z is', 15)
('x << z is', 240)
```

APEC

Assignment operators

- Used to assign values to variables.

Operator	Name	Example	Equivalent to
=	Equal to	x = 10	x = 10
+=	Plus Equal to	x += 10	x = x + 10
-=	Minus Equal to	x -= 10	x = x - 10
*=	Multiplication Equal to	x *= 10	x = x * 10
/=	Division Equal to	x /= 10	x = x / 10
%=	Modulus Equal to	x %= 10	x = x % 10
//=	Float Division Equal to	x //= 10	x = x // 10
**=	Exponent Equal to	x **= 10	x = x ** 10
&=	AND Equal to	x &= 10	x = x &10
|=	OR Equal to	x |= 10	x = x | 10
^=	XOR Equal to	x ^= 10	x = x ^ 10
>>=	Right shift Equal to	x >>= 2	x = x >>2
<<=	Left shift Equal to	x >>= 2	x = x >>2

APEC

Membership operators

- Used to test whether a value or variable is found in a set.
- Two membership operators: **in, not in.**

Operator	Example	Explanation
in	x in y	True if value/variable is found in the sequence
not in	x not in y	True if value/variable is not found in the sequence

Example:

```
a = "hello"
print ("h" in a)
print ("x" not in a)
```

Output:

```
True
True
```

Membership - Example

```
x = 'Python Program'
y = {1:'John',2:'9090'}
print('y' in x)
print('p' in x)                    #case sensitive
print('hello' not in x)
print(1 in y)
print('a' in y)
```

Output:

```
True
False
True
True
```

APEC

False

APEC

Identity operators

- Compares the memory locations of two objects.

Operator	Example	Explanation
is	x is True	True if the operands are identical ie. they refer to the same object
is not	x is not True	True if the operands are not identical ie. they do not refer to the same object

```
a = 10
b = 10
print (a is b)
a = 20
print (a is b)
print (a is not b)
```

Output:

```
True
False
True
```

APEC

Identity - Example

```
x1 = 5
y1 = 5
x2 = 'Python'
y2 = 'python'
x3 = [1,2,3]
y3 = [1,2,3]


print(x1 is y1)          # x1 and y1 are integer comparison
print(x1 is not y1)
print(x2 is y2)          #case sensitive, x2 and y2 are string
comparison
print(x3 is y3)          # lists cannot be compared
```

Output

```
True
False
False
False
```

APEC

Boolean operator

- A boolean expression or logical expression evaluates to one of two states, either True or False.

APEC

Boolean Values

```
#Python program to add Boolean numbers
a = True
b = False
c = 20
print (a+b+c)
```

Output

21

Boolean values, True means 1 and False means 0

APEC

Example

```
#python program
result = True + 312 + 1.34
print (result)
result = False + 312 + 1.34
print (result)
result = False + 312 + 1.34j
print (result)
```

Output

314.34

313.34 (312+1.34j)

APEC

Python Operator Precedence

- **Precedence:**
  - Defines the priority of an operator
- **Associativity:**
  - If two operators have equal precedence, then evaluate the expression from left to right (or) right to left.

APEC

Precedence Of Operators

Operator	Description
**	Exponentiation
~ + -	Complement, unary plus and minus
*, /, %	Multiplication, division, remainder
+, -	Addition, subtraction
<<, >>	Bitwise shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, =, >, >=,<>, !=, ==	Comparisons, membership, identity
not x	Boolean NOT
and	Boolean AND
or	Boolean OR

← Highest Precedence

← Lowest Precedence

APEC

lambda	Lambda expression

APEC

Most frequent used

- **PEDMAS**
  - Parenthesis
  - Exponentiation
  - Division & Multiplication
  - Addition & Subtraction

  - Operators with the same precedence are evaluated left to right.
  - 2+3*4 = 14
  - 3-2+1 = 2

  - <u>Note:</u>
    - Inside a expression, unary his given importance.

APEC

- **After parenthesis, check for unary operator**

APEC

Example

```
#python program to explain precedence
print ("2+6*5 = ", 2+6*5)
# * have higher precedence than +
print ("2+(6*5) = ", 2+(6*5))
print ("(2+6)*5 = ", (2+6)*5)

print ("2*16/2 = ", 2*16/2)
print ("2*(16/2) = ", 2*(16/2))
print ("(2*16)/2 = ", (2*16)/2
```

Output

```
('2+6*5 = ', 32)
('2+(6*5) = ', 32)
('(2+6)*5 = ', 40)
('2*16/2 = ', 16)
('2*(16/2) = ', 16)
```

APEC

('(2*16)/2 = ', 16)

APEC

Example

```
#python program to explain precedence
a = 200
b = 100
c = 150
d = 50
result = a + b * c / d
print ("Result = ", result)
result = (a + b) * c / d
print ("Result = ", result)
result = (a + b) * (c / d)
print ("Result = ", result)
result = a + (b * c) / d
print ("Result = ", result)
result = (a + (b * c)) / d
```

Output

APEC

Output

```
('Result = ', 500)
('Result = ', 900)
('Result = ', 900)
('Result = ', 500)
('Result = ', 304)
```

APEC

```
#Python Program to explain Precedence
a = 20
b = 10
c = 15
d = 5
e = 0

e = a + b * c / d
print "Value of a + b * c / d is ",  e
e = (a + b) * c / d
print "Value of (a + b) * c / d is ",  e
e = a + (b * c) / d
print "Value of a + (b * c) / d is ",  e
e = (a + (b * c)) / d
print "Value of (a + (b * c) / d is ",  e
e = ((a + b) * c) / d
print "Value of ((a + b) * c) / d is ",  e
e = a + b * (c / d);
print "Value of a + b * (c / d) is ",  e
e = (a + b) * (c / d);
```

Output

```
Value of a + b * c / d is  50
Value of (a + b) * c / d is  90
Value of a + (b * c) / d is  50
Value of (a + (b * c) / d is  34
Value of ((a + b) * c) / d is  90
Value of a + b * (c / d) is  50
Value of (a + b) * (c / d) is  90
```

APEC

```
print "Value of (a + b) * (c / d) is ",  e
```

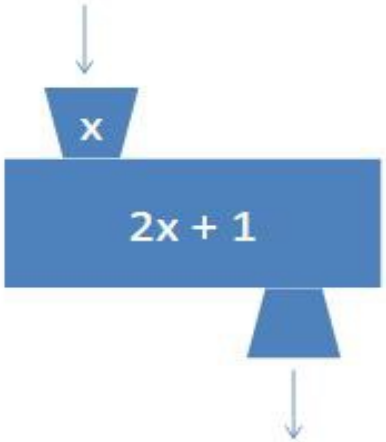
APEC

Function - Intro

A function is a machine



- You give it input
- It produces a result

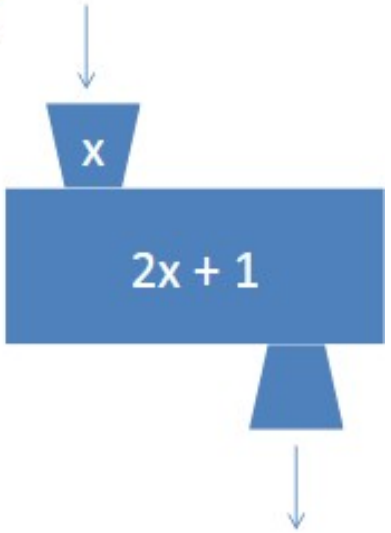


In math: func(x) = 2x + 1

Intro - contd...

Creating a function

Define the machine,
including the input and the result




```
def func(x):
    return 2*x + 1
```

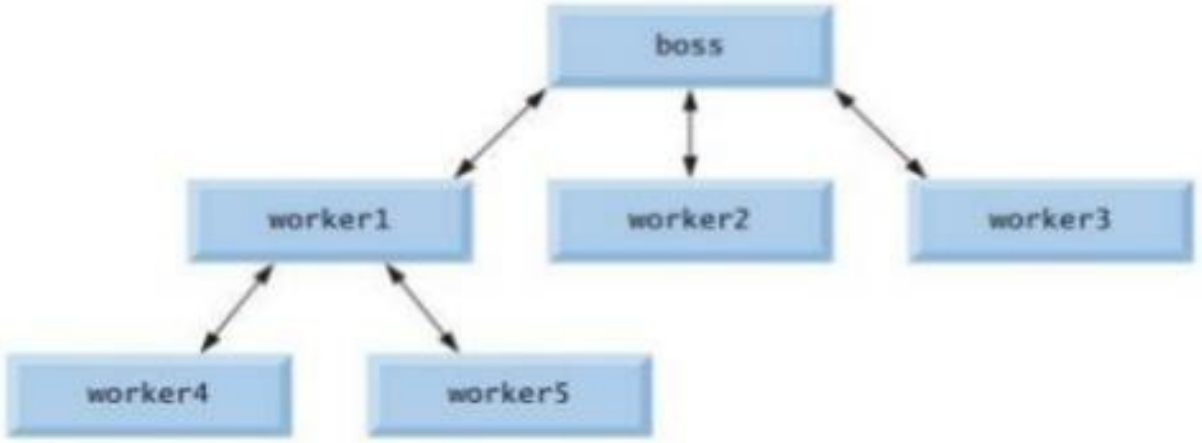
Divide And Conquer

- The Top-down design approach is based on **dividing** the **main problem** into **smaller tasks which** may be **divided** into **simpler tasks**, then implementing **each simple** task by a subprogram or a **function**.
- This technique is called **divide and conquer**.
- **Example:** Construction of a house

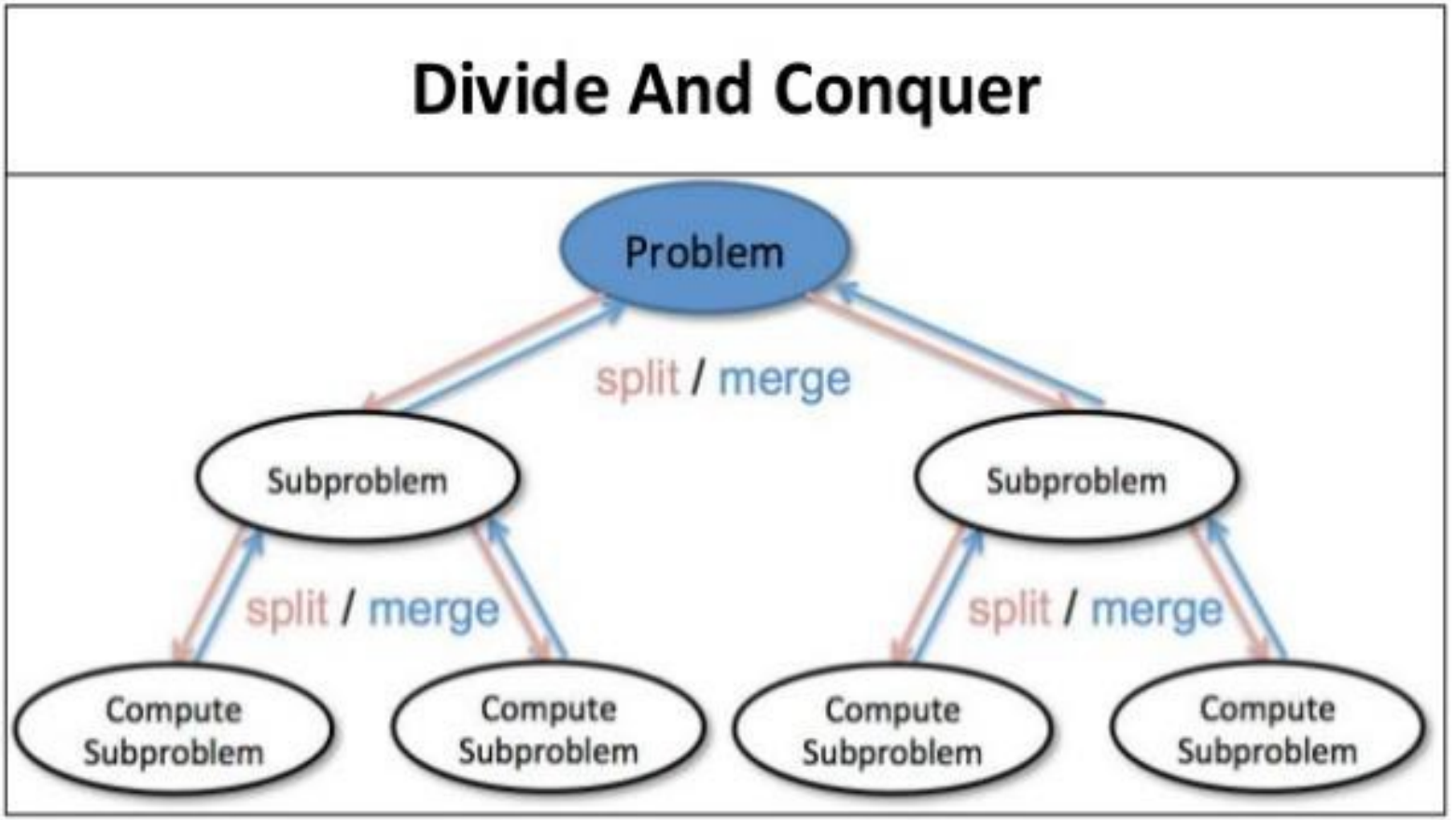


Divide And Conquer

- Below figure shows the boss function communicating with several worker functions.
- The **boss function divides** the **responsibilities** among the worker functions, and worker1 acts as a " boss function" to worker4 and worker5.



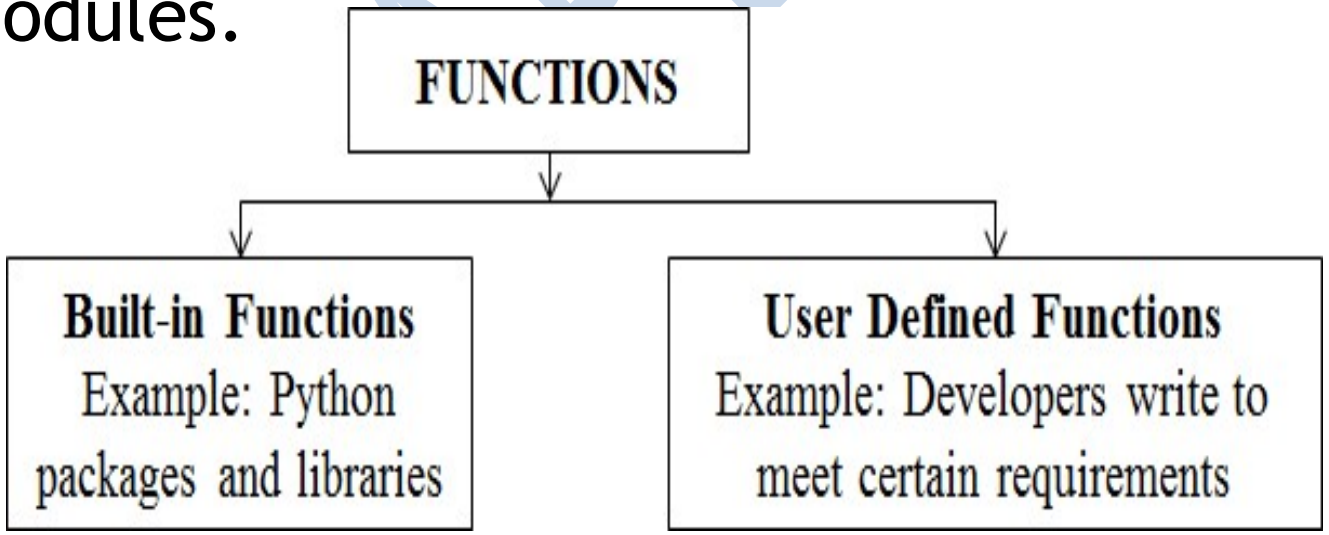
Divide And Conquer



Methods are Used for Divide a Large Code into Modules

Python Functions

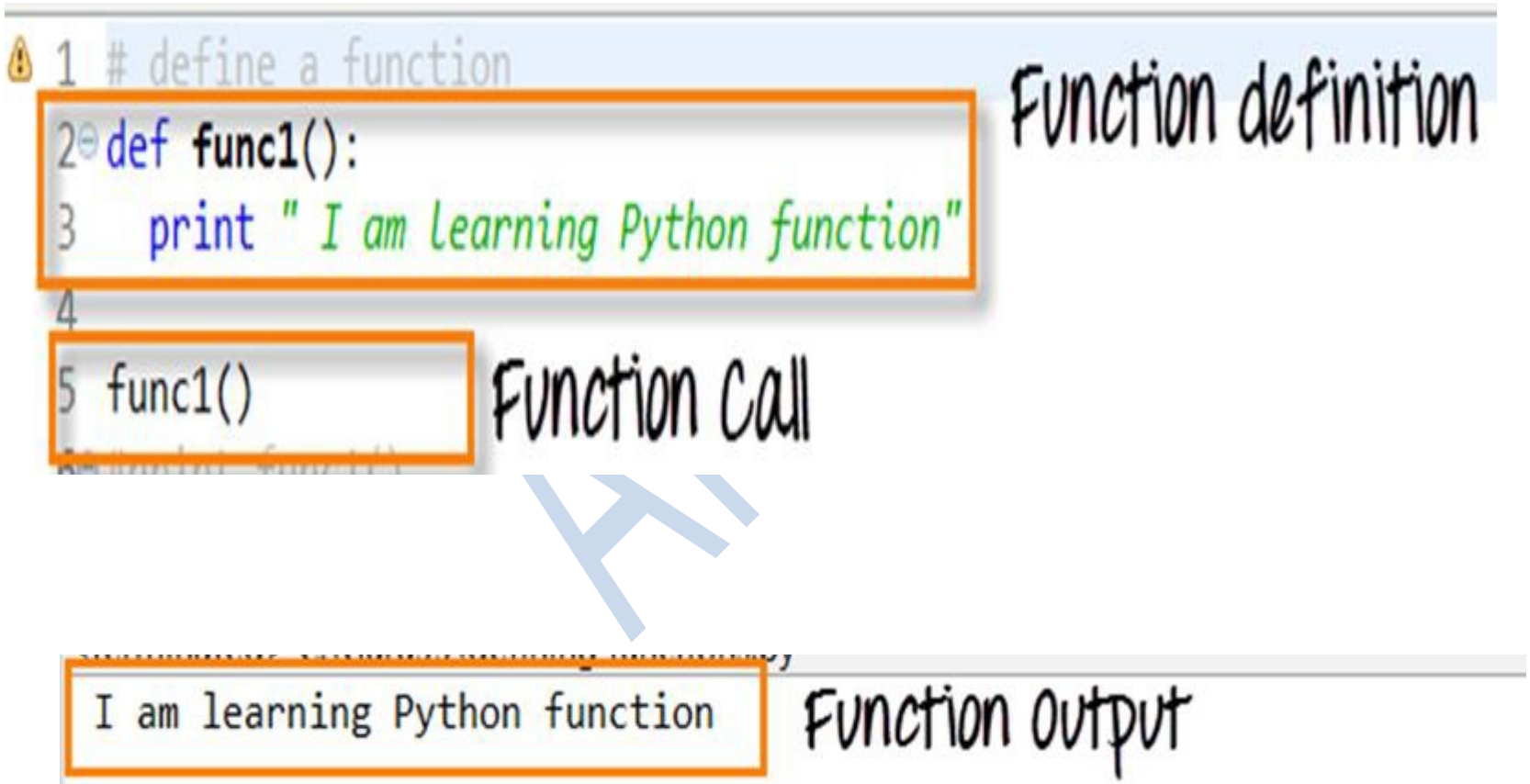
- Functions are a block of related statements to perform a specific task. Functions help to break the program into small manageable units or modules.



Types of Functions in Python

APEC

Example


1 # define a function
2 def func1():
3 print " I am learning Python function"
4
5 func1()
Function definition
Function Call
I am learning Python function
Function Output

Example

Function definition

```
def test_fun():
    print ("this is test function")
    return

test_fun()
```

Function call

Output:
this is test function

APEC

Python Indentation

- Python uses indentation **(i.e., 4 whitespaces)** to define block of code.

```
def test_fun():
# indented four spaces
    print("x is 1.")

test_fun()
```

Execution:

```
sh-4.3$ python program2.py
x is 1.
```

APEC

Rules to define a function in Python

- Every function blocks begin with the keyword **def** followed by the function name and parentheses ( ).
- Input parameters (arguments) through which we pass values to a function are defined inside these parentheses.
- A colon (:) is used to mark the end of function header
- Optional documentation string (to describe what the function does.
- One or more valid python statements make up the function body and all statements must have same indentation level (usually 4 spaces).
- An optional return statement to return a value from the

function.

APEC

Syntax of Function


```
def  function_name(parameters):
 """docstring"""
    Statement(s);
    return
```

```
def my_own_print(line):
    "This function prints a line"
    print(line)
    return

my_own_print("This is my program")
```

Function Definition

Function call

APEC

APEC

Example


```
my_own_print("This is my program")

def my_own_print(line):
    "This function prints a line"
    print(line)
    return
```

APEC

```
Traceback (most recent call last):
  File "C:/Python27/p1.py", line 1, in <module>
    my_own_print("This is my program")
NameError: name 'my_own_print' is not defined
```

Example


```
def my_own_print(fline):
    "This function prints a line"
    print(fline)
    return

line = "This is my program"
my_own_print(line)
```

APEC

Output:

This is my program

Formal parameter vs actual parameter

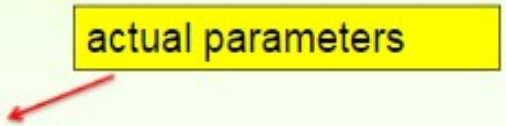
APEC

Testing a function

- function must be defined before you use it
- To use it:




```
def addit(parm1, parm2):
    ans=parm1 + parm2
    return ans
```




```
mysum = addit (5,2)
print ("result of function call is = ", mysum)
```

Example – Formal and actual parameters same

```
def my_details(name,age,salary):
    "This function prints a line"
    print("My name: ", name)
    print("My age: ", age)
    print("My salary: ", salary)
    return

name = input("Enter your name :")
age = int(input("Enter your age :"))
salary = float(input("Enter your salary :"))
```

Output

Output

```
Enter your name : Hari
Enter your age :23
Enter your salary :8902
('My name: ', ' Hari')
('My age: ', 23)
('My salary: ', 8902.0)
```

my_details(name,age,salary)

APEC

Formal and actual parameters different

```
def my_details(fname,fage,fsalary):
    "This function prints a line"
    print("My name: ", fname)
    print("My age: ", fage)
    print("My salary: ", fsalary)
    return

name = raw_input("Enter your name :")
age = int(input("Enter your age :"))
salary = float(input("Enter your salary :"))
```

name,age,salary

Output

```
Enter your name : hari
Enter your age : 45
Enter your salary :79891
('My name: ', ' hari')
('My age: ', 45)
('My salary: ', 79891.0))
```

APEC

my_details(name,age,salary)

APEC

Example – Two functions

```
a = 10
b = 20
def add(a,b):
    print ("Addition =", a + b)
    return

add(a,b)

def sub(a,b):
    print ("Subtraction =", a - b)
    return
```

sub(a,b)

Output

APEC

Output

```
('Addition =', 30)
('Subtraction =', -10)
```

APEC

Example - Nested functions

```
a = 10
b = 20
def add(a,b):
    def sub(a,b):
        print ("Inside nested function =", a - b)
        return
    sub(a,b)
    print ("Outside nested =", a + b)
    return
```

Output

```
('Inside nested function =', -10)
('Outside nested =', 30)
```

Inner Loop

Inner Loop

add(a,b)

Outer Loop

APEC

Outer Loop

APEC

Local and global variables

- **Global variables**
  - Declared outside any function, and they can be used anywhere.
- **Local variables**
  - Declared inside a function, and can be used only inside that function.
  - Note:
    - It is possible to have local variables with the same name in different functions.

APEC

Global variables

Global variables

Global variables

Variable defined outside a function or class is a global variable.

```
counter = 10
def printVar():
    print counter

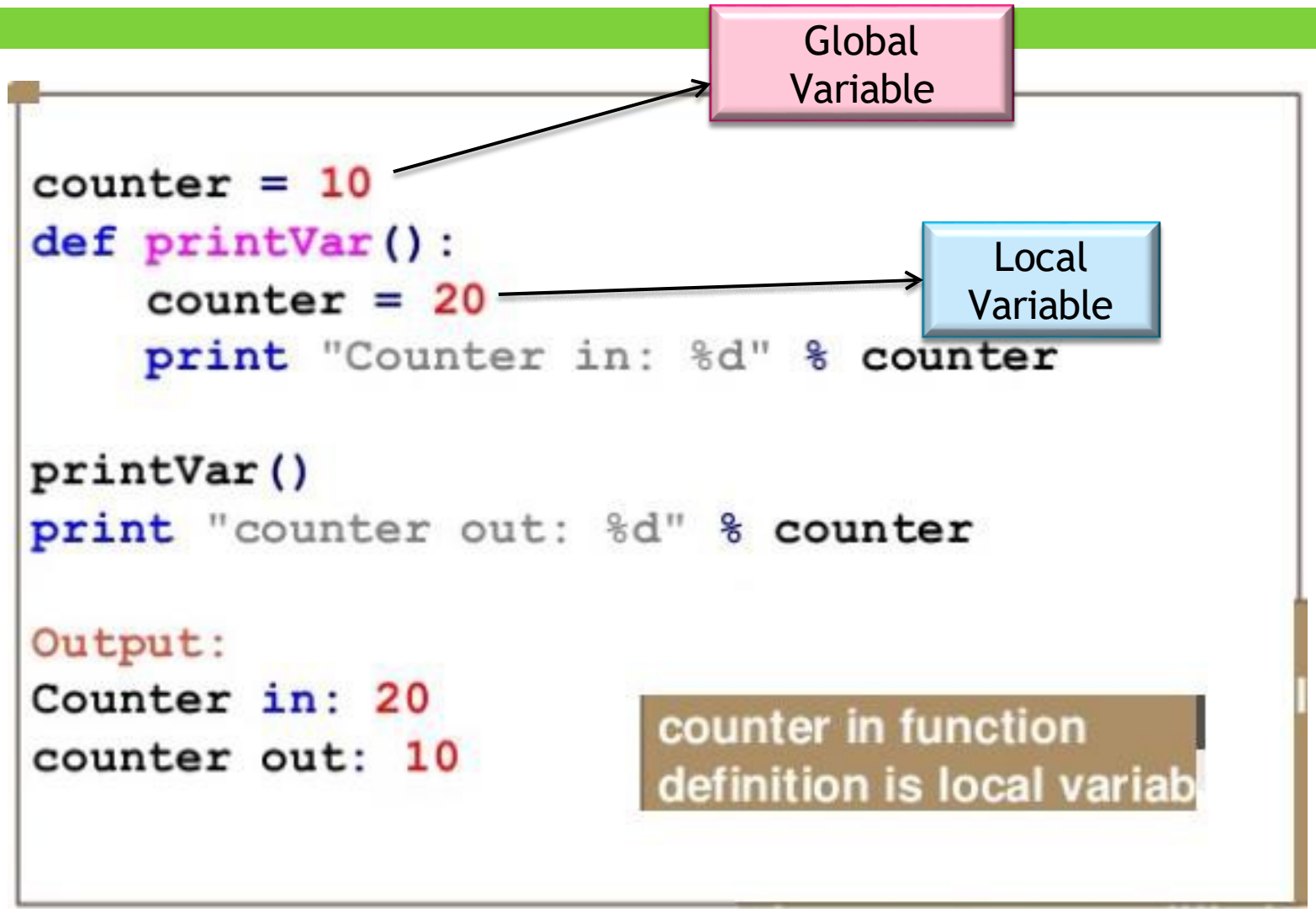
printVar()
print "counter: %d" % counter
```

Counter is a global variable

Output:

```
10
counter: 10
```

Local variable



Example – Scope of the variable

Output

```
a = 10
b = 20
def add(a,b):
```

a,b are the Global variables

Output

```
('Addition inside function = ', 30)

Traceback (most recent call last):
NameError: name 'result' is not defined
```

APEC

```
def add(a,b):
    result = a + b
    print ("Addition inside function = ", result)
    return

add(a,b)
print ("Addition outside function = ", result)
```

result is the Local variable

APEC

Example – Scope of the variable

Example – Scope of the variable

```
a = 10
b = 20
def add(a,b):
    global result
    result = a + b
    print ("Addition inside function = ", result)
    return

add(a,b)
```

a,b are the Global variables

result is also the global variable

Output

```
('Addition inside function = ', 30)
('Addition outside function = ', 30)
```

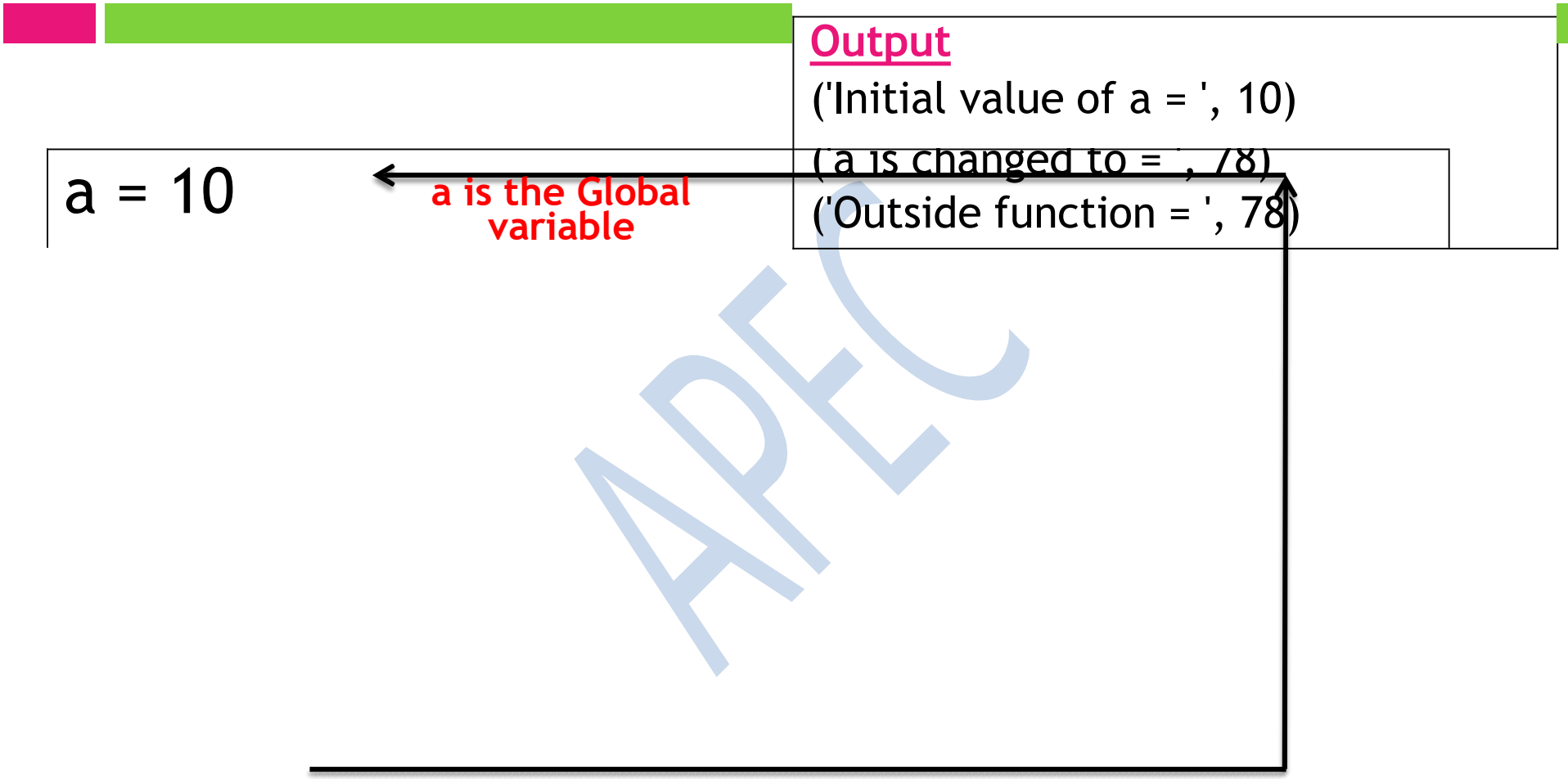
APEC

```
print ("Addition outside function = ", result)
```

APEC

Example – Scope of the variable

Output



APEC

```
def fun():
    global a
    print ("Initial value of a = ", a)
    a = 78
    print ("a is changed to = ", a)
    return

fun()
print ("Outside function = ", a)
```

APEC

Example

```
def add(a,b):
    result = a + b
    print ("Addition (Inside Function) = ", result)
    return result

result = 1000 + add(100,200)
print ("Addition(Outside function) = ", result)
```

APEC

Output

```
('Addition (Inside Function) = ', 300)
('Addition(Outside function) = ', 1300)
```

APEC

Scope and Lifetime of variables

- Variables that are defined inside a function body have a **local scope**, and those defined outside have a **global scope**.
- The lifetime of variables inside a function live, is as long as the function executes. They are destroyed once its returned from the function.

```
    mark1 = 86
    mark2 = 76
    mark3 = 89
    total_marks = 0                    #Global variable

    def sum(mark1, mark2, mark3):              #pass by value
        total_marks = mark1+ mark2+mark3
        print "Total Marks obtained(inside function) = ", total_marks

    sum(mark1, mark2, mark3)              # Passing arguments
    print "Total Marks obtained = ", total_marks

Total Marks obtained(inside function) = 251
Total Marks obtained = 0
```

APEC

Function prototypes

- **Function prototypes depends on**
  - Whether arguments is present or not
  - Whether return value is present or not

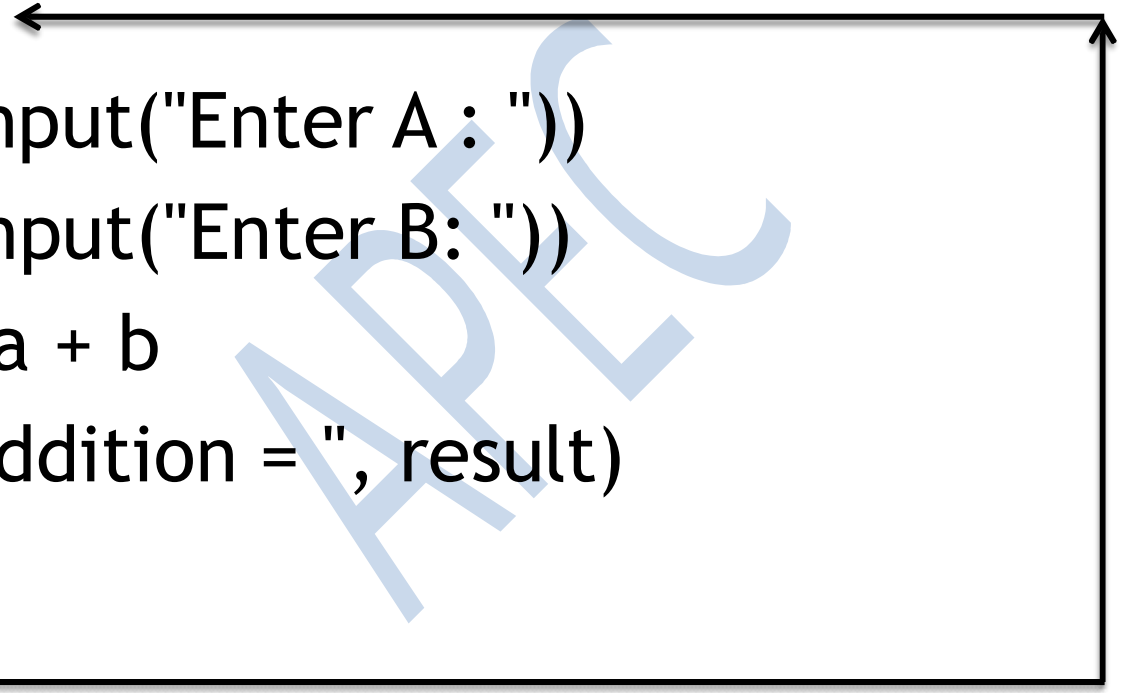
- **Types**
  - Function with no arguments and no return values
  - Function with arguments and no return values
  - Function with no arguments and with return values
  - Function with arguments and with return values

APEC

Function with no arguments and no return values

```
def add():
    a = int(input("Enter A : "))
    b = int(input("Enter B: "))
    result = a + b
    print ("Addition = ", result)

add()
```



Output

Enter A : 90
Enter B: 89

APEC

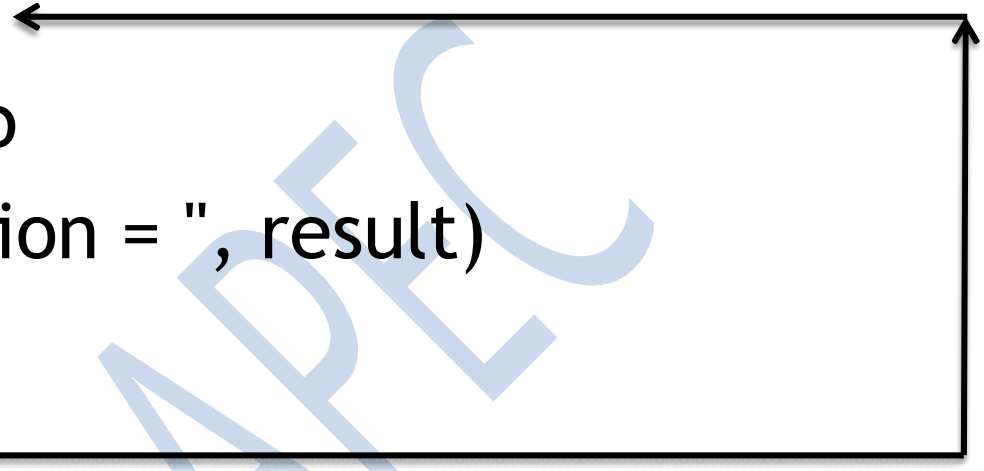
('Additi
on = ',
179)

APEC

Function with arguments and no return values

```
def add(a,b):
    result = a + b
    print ("Addition = ", result)

add(10,30)
```



Output

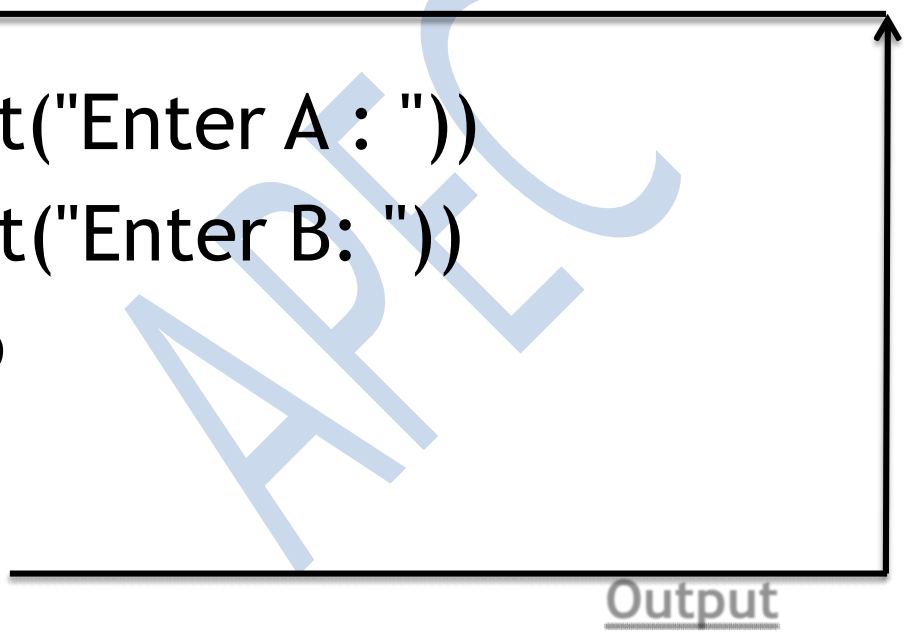
('Addition = ', 40)

APEC

Function with no arguments and with return values

```
def add():
    a = int(input("Enter A : "))
    b = int(input("Enter B: "))
    return a + b


result = add()
```



APEC

```
print ("Addition = ", result)
```

Output

```
Enter A : 78
Enter B: 56
('Addition = ', 134)
```

APEC

Function with arguments and with return values

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

result = add(40,50)
print ("Addition = ", result)
```



Output

('Addition = ', 90)

Note:
def add(a,b):

APEC

```
return a + b
```

APEC

Function Arguments

Following types of formal arguments are used in python:

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length argument or Arbitrary argument

APEC

Required arguments

- In required arguments, the number of arguments passed in the function call should match exactly with the function definition.

Program:

```
def printme( str , n1):
   "This function prints two arguments"
   print str,n1
   return;

printme("Hi", 10)
printme(10, "Hi")
                          #Error
```

Output

```
Hi 10
10 Hi


TypeError: printme() takes
exactly 2 arguments (1 given)
```

APEC

printme(10)

APEC

Keyword arguments

- It allows skipping arguments or placing them out of order because it uses the keywords to match the values with parameters.
- keyword=value.

Program:

```
name = input("Enter name :")
age = int(input("Enter Age :"))

def printme(name,age ):
   print ("My name :", name)
   print ("My age :", age)
   return

printme(name,age)
```

Output

```
Enter name : Vinai
Enter Age : 30
('My name :', ' Vinai')
('My age :', 30)
```

APEC

Contd…

Program:

```
name = input("Enter name :")
age = int(input("Enter Age :"))

def printme(name,age ):
   print ("My name :", name)
   print ("My age :", age)
   return

printme(name = "Sanju",age = 23)
```

Output

```
Enter name : dhaya
Enter Age :23
('My name :', 'Sanju')
('My age :', 23)
```

APEC

Contd…

Program:

```
name = input("Enter name :")
age = int(input("Enter Age :"))

def printme(name,age ):
   print ("My name :", name)
   print ("My age :", age)
   return

printme(name ,age = 23)
```

Output

```
Enter name : vinai
Enter Age :99
('My name :', ' vinai')
('My age :', 23
```

APEC

Default Arguments

Default argument assumes default value if a value is not provided in the function call for that argument.

APEC

Contd…

Program:

```
name = raw_input("Enter name :")
phone = int(input("Enter phone number :"))

def printme(name,phone = 9898756453 ):
   print ("My name :", name)
   print ("My age :", phone)
   return

printme(name,phone)
printme(name)
```

Output

```
Enter name :sanju
Enter phone number :7890563421
('My name :', 'sanju')
('My age :', 7890563421L)
('My name :', 'sanju')
('My age :', 9898756453L)
```

APEC

Variable-length arguments or Arbitrary arguments

- The number of arguments that will be passed into a function may not known in advance.

- Python allows us to handle this kind of situation through function calls with arbitrary number of arguments. Use an asterisk (*) before the parameter name to denote this kind of argument.

APEC

Variable-length arguments or Arbitrary arguments

- The number of arguments that will be passed into a function may not known in advance.

- Python allows us to handle this kind of situation through function calls with arbitrary number of arguments. Use an asterisk (*) before the parameter name to denote this kind of argument.

APEC

Contd…

```
def welcome(*names):
    "This function welcome all people"
    for name in names:
    # names is a tuple with arguments
       print("Hi",name)

welcome("john","jake","leo","jacky")
```

Output

Output

```
('Hi', 'john')
('Hi', 'jake')
('Hi', 'leo')

('Hi', 'jacky')
```

APEC

The Anonymous Functions

Syntax:

lambda arguments:expression

Rules for Lambda function:

- It can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.
- It cannot be a direct call to print because lambda requires an expression
- Lambda functions are single line function. They are not equivalent to inline statements in C or C++.

The Anonymous Functions

```
sum = lambda arg1, arg2: arg1 + arg2;
#Now you can call sum as a function
print "Value of total : ", sum( 70, 20 )
print "Value of total : ", sum( 20, 20 )
```

Output

```
sh-4.3$ python program25.py
Value of total :  90
Value of total :  40
```

The Anonymous Functions

Filter() function in python takes in a function and a list of arguments.

```
# Python program to filter out only even
# elements from a list
list1 = [1,6,5,78,43,32,54,99]
print("List Elements are ", list1)
even_list = list(filter(lambda x:(x%2==0),list1))
print ("Even numbers in the list", even_list)
```

Output

```
('List Elements are ', [1, 6, 5, 78, 43, 32, 54, 99])
('Even numbers in the list', [6, 78, 32, 54])
```

APEC

The Anonymous Functions

map()

map() function in python takes in a function and a list of arguments.

```
# Filter out only even elements from a list

list1 = [1,6,5,78,43,32,54,99]

print("List Elements are ", list1)

even_list = list(filter(lambda x:(x%2==0),list1))

print ("Even numbers in the list", even_list)

# Square of given numbers in a list

square_list = list(map(lambda x:(x*x),list1))

print ("Square of given numbers", square_list)
```

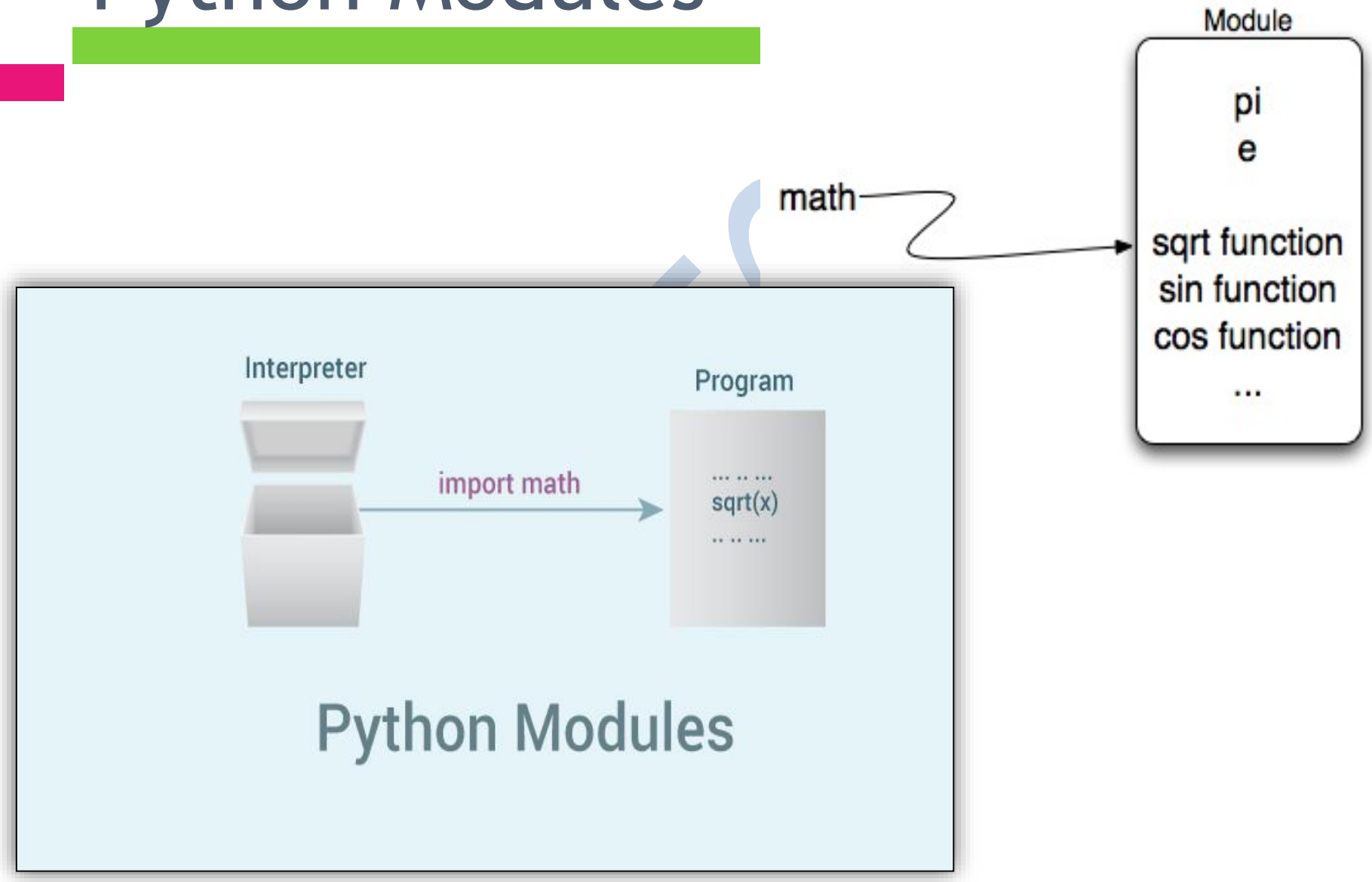
Output

Output

```
('List Elements are ', [1, 6, 5, 78, 43, 32, 54, 99])
('Even numbers in the list', [6, 78, 32, 54])
('Square of given numbers', [1, 36, 25, 6084, 1849, 1024, 2916, 9801])
```

APEC

Python Modules



Python Modules

- Module is a file containing Python definitions and statements.
- It defines functions, classes and variables and includes runnable code also.
- Modules are groups of functions.

APEC

Python Module - Steps

- Create a python file with one or more functions and save it with .py extension
- Include the import statement.
- Using the module name the function is accessed using dot (.) operation

Example Program

Program : *example.py*

```
def add(a, b):
    """This program adds two
    numbers and return the result"""
    result = a + b
    return "Addition of two numbers", result
```

Execution

```
>>> import example
>>>example.add(89,90)
('Addition of two numbers', 179)
```

APEC

Example Program


```
import random
print random.randint(0, 5)          #random integer
print random.random()               #random floating point
print random.random() * 10
my_data = [156, 85, "John", 4.82, True]

#choose an element from the list
print random.choice(my_data)
```

Execution

```
sh-4.3$ python program31.py
1
0.698672804611
5.2069984395
4.82

sh-4.3$ python program31.py
0
0.384566916593
6.85856689662
85
```

APEC

The import Statement

- For a module involving hundreds of functions, Statement is recommended to save loading time.

  Syntax: from python_file import function_name

- Multiple functions can be imported by separating their names with commas.

  Syntax: *from python_file import funtion_name1,*

  Function_name2...

- Use the asterisk symbol(*) to import everything

  Syntax: from python_file import*

APEC

Example Program

Program : *example.py*

```
def add(a, b):
"""This program adds two
   numbers and return the result"""
   result = a + b
   return "Addition of two numbers", result

 def sub(a, b):
"""This program Subtracts two
   numbers and return the result"""
   result = a - b
   return "subtraction of two numbers", result
```

Execution

```
>>> from example import add,sub
>>> add(9,2)
('Addition of two numbers', 11)
>>> sub(9,2)
('subtraction of two numbers', 7)
```

APEC

The Built-in Modules

- There are plenty of built in modules, just as built in functions.The most useful ones are:
  - random
  - math
  - calendar &date-time

Random

- Used to generate random numbers.
- Accepts two parameters: a lowest and a highest number.
- Choose a random element from a set such as a list, called choice.

APEC

Math – (Built in module)

APEC

```
print circle_area(3)
print square_area(3)
```

```
print
rectangle_area(3,4)
```

```
def rectangle_area(width, height):
    # width * height
    return width * height
```

Output:

28.274338823
9
12

Output:

APEC

Math Module

Program:

Normal mode:

```
import math
print("PI value =", math.pi)
```

Execution

```
('PI value =', 3.141592653589793)
```

Interactive mode:

```
from math import pi
```

Renaming

APEC

Interactive mode:

```
>>> from math import pi
>>> pi
3.141592653589793
```

Renaming

```
>>> import math as m
>>> print m.pi
3.14159265359
```

APEC

Math Module

- The math module provides access to mathematical constants and functions.

```
import math
print "Value of Pi = ", math.pi
print "Value of Eulers number = ",math.e
print "Value of 2 radians = ", math.degrees(2)
print "Value of 60 degress = ", math.radians(60)
print "sin(2) radians = ",math.sin(2)
print "cos(0.5) radians = ",math.cos(0.5)
print "tan(0.23) radians = ",math.tan(0.23)
print "Factorial of 5 = ", math.factorial(5)
print "Square Root of 49 = ",math.sqrt(49)
```

APEC

Math – (Built in module)

Program:

```
import math
print("power function =", math.pow(7,2))
print("Factorial of a given number =", math.factorial(5))
print("Absolute number =", math.fabs(590.6890329032903297))
print("Ceil of a number =", math.ceil(590.6890329032903297))
print("Floor of a number =", math.floor(590.6890329032903297))
```

Output:

```
('power function =', 49.0)
('Factorial of a given number =', 120)
('Absolute  number =', 590.6890329032904)
('Ceil of a number =', 591.0)
('Floor of a number =', 590.0)
```

Calendar & date-time

- Python's time and calendar modules help track dates and times.

PROGRAM

```
import calendar
cal = calendar.month(2017, 5)
print "May month calendar of the year 2017:"
print cal

import time
ticks = time.time()
print "Number of ticks since 12:00am, January 1, 1970:", ticks
localtime = time.localtime(time.time())
print "Local current time :", localtime

#Formatted time
localtime = time.asctime( time.localtime(time.time()) )
```

APEC

```
print "Local current time :", localtime
```

APEC

Calendar&date-time

Execution

```
May month calendar of the year 2017:
      May 2017
Mo Tu We ThFr 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


Number of ticks since 12:00am, January 1, 1970: 1494493602.96
Local current time : time.struct_time(tm_year=2017, tm_mon=5, tm_mday=11,
tm_hour=9, tm_min=6, tm_sec=42, tm_wday=3, tm_yday=131, tm_isdst=0)


Local current time : Thu May 11 09:06:42 2017
```

APEC

Some useful links

- https://www.tutorialspoint.com/python/python_online_quiz.htm
- http://www.mypythonquiz.com/question.php
- https://campus.datacamp.com/courses/intro-to-python-for-data-science
- https://www.interviewmocha.com/tests/python-3-coding-test-basic

APC


THANK YOU


THANKS