Unit - IV

4.1 Basic Features of Python

Python ஒரு எளிய, எளிதாக கற்றுக்கொள்ளக்கூடிய, high level programming languageஆகும். இதை Guido Van Rossum 1980 முதல் 1990 வரையிலான காலகட்டத்தில் National Research Institute for Mathematics and Computer Science, Netherrlandsல் உருவாக்கினார். Python ஒரு scripting languageஆகும்.

Features

- Python ஒரு Interpreted Language ஆகும். இதில் intermediate code உருவாக்கப்படுவதில்லை.
- Python ஒரு Object-Oriented Language ஆகும். இதில் class, objects மற்றும் inheritance ஆகிய conceptகலை உபயோகப்படுத்த முடியும்.
- Python எளிதாக புரிந்துகொள்ளக்கூடிய மற்றும் எளிதாக programகளை எழுதக்கூடிய language ஆகும்.
- Pythonல் dynamic type checking support செய்யப்படும். இதன்மூலம் ஒரு variableன் data type தானாக அதில் store செய்யப்படும் valueவை பொறுத்து assignஆகும்.
- Python ஒரு portable language ஆகும். இதை அணைத்து Operating Systemதிலும் இயக்க முடியும்.
- Python ஒரு Scalable lanugage ஆகும். இதன்மூலம் நம்மால் பெரிய programகளை develop செய்து பராமரிக்க முடியும்.

Installing - Windows

IDLE Pythonனின் built-in Integrated Development Environment ஆகும். இதனை www.python.org websiteல் download செய்யலாம்.

Steps to Install

- Download Python 2.7.14 MSI installer for windows.
- Start Installation by double click MSI Installer file.
- In Setup Wizard select installation folder for Python.
- To start Python select IDLE(Python GUI) from start menu.

Installing - Linux

Linux OSல் Python install செய்வதற்கு command lineல் கீழே கொடுக்கப்பட்டுள்ள commandஐ type செய்யவேண்டும்.

```
$ sudo apt-get install idle
```

Installation முடிந்த பிறகு Python applicationஐ start செய்வதற்கு command lineல் idle என்று type செய்யவேண்டும்.

```
$ idle
```

IDLE - Features

- Cross - Platform: IDLE அணைத்து OSகளிலும் execute செய்யலாம்.
- Interactive Interpreter: IDLE நாம் program type செய்யும்பொழுது syntax highlighting மற்றும் code completion போன்ற அம்சங்களை கொண்டிருக்கும்.
- Multi-Window: IDLEல் ஒரே நேரத்தில் ஒன்றுக்கும் மேற்பட்ட script windowவினை open செய்யமுடியும்.
- Basic text Operations: IDLEல் cut, copy, paste மற்றும் find போன்ற text operationகலை செய்யமுடியும்.
- Debugger: IDLEல் ஒரு programஐ ஒரு ஒரு lineஆக execute செய்தல் மற்றும் break pointன் மூலமாக program executionஐ கண்காணிக்க முடியும்.
- Interactive Mode: இந்த முறையில் Python Script நேரடியாக interpreter windowவில் type செய்யப்பட்டு execute செய்யப்படுகிறது.
- Script Mode: இந்த முறையில் Python Scriptஐ .py எனும் fileல் save செய்தபிறகு interpreterல் execute செய்யப்படுகிறது.

4.2 Variables and Strings

Pythonல் variableகள் மாறக்கூடிய மதிப்புகளை சேமிக்க உதவுகிறது. C programல் உள்ளதுபோல variableகளுக்கான data typeஐ முன்கூட்டியே pythonல் நிர்ணயிக்க வேண்டியதில்லை. Pythonல்ஒருvariableலின் data typeஆனது அதில் சேமித்துள்ள மதிப்பை பொறுத்து தானாக நிர்ணயிக்கப்படுகிறது.

Nameing Rules

- முதல் எழுத்து A-Z அல்லது a-z அல்லது underscore(_)ஆக இருக்கவேண்டும்.
- மற்ற எழுத்துக்கள் character, underscore மற்றும் nunmber (0-9) இவைகளின் சேர்கையாக இருக்கலாம்.
- Keywordகளை variable nameஆக கொடுக்க முடியாது.

Example

```
x = 10
y = 20
_sum = x + y
print "Result is ",_sum," by adding ",x,y
```

Strings

String என்பது characterகளின் தொகுப்புஆகும். Pythonல் string valueகளை single quotes(‘) அல்லது double quotes(“)க்லைன் மூலமாக உருவாக்கலாம். ஒன்றுக்கும் மேற்பட்ட வரிகளில் எழுதப்படும் string valueகளை triple quotes(“““) மூலமாக உருவாக்கலாம்.

Example:

```
name = ‘Mani’
course = “Computer”
college = “““Tamilnadu Polytechnic College
          Madurai 11”””
```

Data Types

Data Type என்பது ஒரு variableலில் சேமித்திருக்கும் valueவை பற்றியும் அதில் நாம் எந்தவிதமான operationகளை செய்யமுடியும் என்பதை தீர்மானிக்கும். Pythonல் ஆறு விதமா data typeகள் உள்ளன அவை

1. Numeric
2. List
3. Tuple
4. String
5. Set
6. Dictionary

Numeric

Pythonல் numeric data type number valueகளை சேமிக்கபயன்படுகிறது. Numeric data typeல் நான்கு விதமான valueகளை சேமிக்க முடியும்

Integer - இதில் முழு எண்கள் store செய்யப்படுகிறது.

Example

```
#integer
a = 10
b = 20
res = a + b
print "Result",res
```

Long Integer - இதில் அதிக மதிப்புடைய integer valueகளை store செய்ய முடியும்

Example

```
#long integer
a = 1234567890
b = 1987654321
res = a + b
print "Long Int Result", res
```

Float - இதில் decimal valueகளை store செய்யலாம்.

Example

```
#float
a = 1.23
b = 3.45
res = a + b
print "Float Result", res
```

Complex - இதில் complex number (2+3i) store செய்யலாம். Pythonல் complex numberஐ குறிக்க complex() function அல்லது j பயன்படுத்தபடுகிறது

Example

```
#complex numbers
a = complex(2,3)
b = 1+4j
res = a + b
print res
```

List

List என்பது வரிசையாக பல valueகளை ஒரேvariable nameல் store செய்ய உபயோகிக்கப்படுகிறது. இது square bracket ([]) மூலம் உருவாக்கப்படுகிறது. இதில் வேறுவேறு data typeகளை store செய்யலாம்.

Example

```
#list
_mylist = [110,10.43,1+3j,'hello']
print _mylist
```

Tuple

Tuple என்பது வரிசையாக பல valueகளை ஒரேvariable nameல் store செய்ய பயன்படுகிறது . இது parantheses (( )) மூலம் உருவாக்கப்படுகிறது. இது ஒரு read only list ஆகும். இதில் கொடுக்கப்பட்ட valueகளை மாற்றமுடியாது.

Example

```
#Tuple
_mytuple = (100,10.43,1+3j,'hello')
print _mytuple
#tuple read only
#_mytuple[1] = 12.43
```

String

String என்பது characterகளின் தொகுப்புஆகும். Pythonல் ஒரு stringல் உள்ள characterகளை arrayவை access செய்வதுபோல indexஇன் மூலமாக access செய்யலாம்.

Example

```
#string
a = 'Welcome'
b = "to Python. "
print a
print b
#To Print 4th Character
print a[3]
```

Set

Set என்பது unordered collectionஆகும். Unordered Collection என்பதை index valueவின் மூலமாக access செய்ய முடியாது. இதை உருவாக்க braces ({}) உபயோகிக்கப்படுகிறது.

Example

```
#set
seta = {11,23,11,23,45,1}
print seta
```

Dictionary

Dictionary என்பது key-value pairன் unordered collectionஆகும். இதனை உருவாக்க braces ({}) உபயோகிக்கப்படுகிறது. இதில் அணைத்து elementகளும் Key-Value pairஆக (Key:Value) கொடுக்கப்பட்டிருக்கும். Keyஐ indexஆக பயன்படுத்தி value பெறப்படுகிறது.

Example

```
#Dictionary
emp = {100:'Balaji',102:'Anjay',103:'Xavier'}
print emp
print emp[102]
```

Operators

Operators என்பது mathematical அல்லது logical operationகளை செய்யபயன்படும் symbolகள் ஆகும்.

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

Arithmetic Operators

இந்த operatorகள் numeric valueகளை கொண்டு ஒரு numeric valueவை return செய்யும்.

Symbol	Operator Name	Description
+	Addition	add two values
-	Subtraction	subtract two values
*	Multiplication	multiplies two values
/	Division	divides two values
%	Modulus	return remainder of dividing two values

Assignment Operators

Assignment Operator (=) என்பது வலது பக்கத்தில் உள்ள valueவை இடது பக்கத்தில் உள்ள variableக்கு assign செய்ய உபயோகிக்கப்படுகிறது.

Symbol	Operator Name	Description
=	Equal	assign value from right to left side variable
+=	Add and	adds value on right to left and store in left side variable
*=	Multiply and	multiply value on right to left and store in left side variable
/=	Divide and	divide value on right to left and store in left side variable
%=	Modulus and	return remainder of value on right to left and store in left side variable

Comparison (Relational) Operators

Comparison Operators இரண்டு valueகளை ஒப்பிட பயன்படுகிறது.

Symbol	Operator Name	Description
==	Double Equal	check if two values are same
<>	Not equla to	check if two values are not same
>	Greater than	check if left value is greater than right value
<	less than	check if eft value is less than right value
<=	less than equal to	check if left value is less than or equal to right value
>=	greater than equal to	check if left value is greater than or equal to right value

Logical Operators

Logical Operators ஒன்றுக்கும் மேற்பட்ட condtionகளை ஒரு decision statementல் உபயோகிக்க பயன்படுகிறது.

Symbol	Operator Name	Description

or	Logical or	if any one condition is true then result is true
and	logical and	if both conditions are true then result is true
not	logical not	it reverse the result

Bitwise Operators

Bitwise Operators binary valueல் கணக்கிட உதவுகிறது.

Symbol	Operator Name	Description
&	Binary AND	if both are 1 will return 1
|	Binary OR	if any one value is 1 will return 1
^	Binary XOR	will return 1 if both value are different and will return 0 if both value are same
<<	Binary Left Shift	moves binary value of left value by number given in right value in left direction
>>	Binary Right Shift	moves binary value of left value by number given in right value in right direction

Identity Operators

Identity Operators இரண்டு variableகளின் memory locationஐ compare செய்ய பயன்படுகிறது.

Symbol	Description	Example

is	return true if both variable point to same memory	#IS
		x = 10
		y = x
		print 'x is y', (x is y)

is not	return true if both variable point to different memory	#IS NOT
		x = 10
		y = 20
		print 'x is y', (x is not y)

Membership Operators

Membership Operator ஒரு value மற்றொரு sequenceல் (string, list, tuple) இருக்கிறதா என்பதை சோதிக்கிறது.

Symbol	Description	Example

in	return true if both the given value is found in the list	#in
		x = 'Hello world'
		print 'H in Hello world', ('H' in x)

not in	return true if both the given value is not found in the list	#not in
		x = 'Hello world'
		print 'A in Hello world', ('A' not in x)

Input/Output Functions

Input

Python scriptல் userரிடம் இருந்து valueகளை பெறுவதற்கு input function உபயோகிக்கப்படுகிறது.

syntax

variable = input(‘Prompt Message’)

Example

```
#input from command line
a = input('Enter a:')
b = input('Enter b:')
res = a + b
print 'Result is', res
```

Output

Python Scriptல் இருந்து valueகளை userக்கு display செய்வதற்கு print function உபயோகிக்கப்படுகிறது. Print function இரண்டு விதமாக outputஐ கொடுக்கும் அது unformatted output அல்லது formatted outputஆகும். Unformatted outputஐ display செய்வதற்கு print functionல் coma(,) உபயோகிக்கப்படுகிறது.

Syntax

```
print 'Message',value
```

Formatted outputஐ display செய்வதற்கு format function பயன்படுகிறது. இதில் உள்ள valueகளை index numberஐ கொண்டு print functionல் display செய்கிறது.

Syntax

```
print 'Message {0}'.format(value)
```

Example

```
#Unformatted and Formatted Output
a = 10
b = 20
res = a + b
#Unformatted
print 'sum of',a,'+',b,' =',res
#Formatted
print 'sum of {1} + {0} = {2}'.format(a,b,res)
```

Decision Control-Conditional Statements

Conditional Statements என்பது ஒரு குறிப்பிட்ட conditionஐ execute செய்து, அதன் மதிப்பை (TRUE/FALSE) பொறுத்து எந்த statementஐ execute செய்யவேண்டும் என்பதை நிர்ணயிக்கும். Python languageல் decision control செய்வதற்கு if statement பயன்படுத்தப்படுகிறது.

- Simple if statement
- If .. else statement
- if .. elif statement

Simple if Statement

இந்த conditional statementல் கொடுக்கப்பட்ட condition TRUEஆக இருந்தால் execute செய்யவேண்டிய statement மட்டும் இடம்பெற்றிருக்கும்.

Syntax

```
if condition:
        statements
```

Example

```
#simple if statement
age = input("Enter Your Age: ")
if age >= 18:
        print "voting age"
print "You are allowed to vote"
```

if..else statement

இந்த conditional statementல் இரண்டு பகுதிகள் இருக்கும். கொடுக்கப்பட்ட condition TRUEஆக இருந்தால் execute செய்யவேண்டிய statementகள் if statementக்கு அடுத்தும் condition FALSEஆக இருந்தால்

execute செய்யவேண்டிய statementகள் else statementக்கு அடுத்தும் கொடுக்கப்படும்.

Syntax

```
if condition:
        TRUE Statements
else :
        FALSE Statements
```

Example

```
#if..else statement
age = input("Enter Your Age: ")
if age >= 18:
        print "Old enough"
        print "You are allowed to vote"
else:
        print "You are young"
        print "Not allowed to vote"
```

if .. elif statement

ஒன்றுக்கும் மேற்பட்ட conditionகளை வரிசையாக check செய்வதற்கு இந்த statement பயன்படுத்தப்படுகிறது

Syntax

```
if condition1:
        statements
elif condition2:
        statements
elif condition3:
        statements
else:
        statements
```

Example

```
#if..elif statement
age = input("Enter Your Age: ")
if age >= 1 and age <= 5:
        print "Child"
        print "No Ticket Required"
elif age >= 6 and age <= 12:
        print "Child"
        print "1/2 Ticket Required"
elif age >= 12 and age <= 60:
        print "adult"
        print "Full ticket"
else:
        print "Senior Citizen"
        print "20% Concession"
```

Loops

ஒரு குறிப்பிட்ட code blockஐ மீண்டும் மீண்டும் பலமுறை execute செய்வதற்கு looping statements பயன்படுத்தப்படுகிறது. Python languageல் இரண்டுவிதமான looping statementகள் உள்ளன அவை.

- While Loop
- For Loop

While Loop

இந்த loop statementல் முதலில் conditionஐ check செய்யப்படும் அந்த condition TRUEஆக இருக்கும்வரை loopல் கொடுக்கப்பட்டிருக்கும் statementகள் executeஆகும். Condition FALSE ஆனதும் loop execution நிறுத்தப்படும்.

Syntax

```
while condition:
        statements
```

Example

```
#while loop statement
print "Sum of the series"
num = input("Enter a number: ")
x = 1
ser = 0
while x <= num:
        ser = ser + x
        x += 1
print ('sum of the series 1 to {0} is {1}'.format(num,ser))
```

For Loop

For loop statement conditionஐ check செய்வதற்கு பதிலாக ஒரு sequenceல் உள்ள elementகளை கொண்டு loopஐ execute செய்யும். Sequence என்பது list, tuple, set அல்லது range() functionஆக இருக்கும்.

Syntax

```
for element in sequence:
        statements
```

Example1:

```
#for loop statement
print "sum of element's in a Tuple"
tup = (1,2,3,12,43,10,4)
res = 0
for x in tup:
        res = res + x
print "sum of Tuple is",res
```

Example2:

```
#For loop using range
a = input("Enter a number")
for i in range(0,a):
        print '*',
```

4.3 Sequences:

Lists: Introduction

Pythonல் list data type பலஇடங்களில் எளிதா உபயோகப்படுத்தக்கூடிய ஒரு sequenceஆகும். இதை உருவாக்க square bracketல் comma-separated valueகளாக கொடுக்கப்படும். ஒரு empyt listஐ உருவாக்க கீழே கொடுக்கப்பட்டுள்ள syntax உபயோகிக்கப்படுகிறது

Syntax

```
lst = []
lst = list()
```

Valueகளுடன் கூடியlistஐ உருவாக்க இந்த syntax உபயோகிக்கப்படுகிறது

Syntax

```
lst = list(object)
lst = [val1, val2, val3…]
```

Listல் உள்ள valueகளை access செய்வதற்கு index value பயன்படுத்தப்படுகிறது. Index valueவை உபயோகித்து listல் உள்ள valueவை மாற்ற முடியும்.

Example

```
lst_a = [1,2,'welcome',4]
print "Second Element: ",lst_a[1]
lst_a[0]='hai'
print lst_a
```

List Methods

Listல் கீழே கொடுக்கப்பட்டுள்ள methodsஐ பயன்படுத்தலாம்.

Method	Description
list.count(item)	Count number of occurrece of *item*
list.append(item)	Add *item*value at last of the list
list.insert(index,item)	Add *item*at the given *index* position
list.index(item)	Get index position of *item*
list.pop()	remove an *item* from end of the list
list.remove(item)	Remove the given *item* from the list
list.reverse()	Reverse the list content
list.sort()	Sort the list content

Fixed size lists and arrays

ஒரு list உருவாக்கும் பொழுது அதன் sizeஐ முன்னதாக கொடுத்து உருவாக்குவதற்கு pythonல் repetition operator (*) மற்றும் None value பயன்படுத்தப்படுகிறது.

Example

```
a = [None] *3
print a
a[0] = 3
a[1] = 1
a[2] = 5
print a
```

Output

```
[None, None, None]
[3,1,5]
```

Pythonல் இந்த முறையை பயன்படுத்தி two dimensional arrayஐ எளிதாக உருவாக்கமுடியும்.

Syntax

```
variable = [None] * rowSize
variable[row index] = [None] * columnSize
variable[row index][col index]
```

Example

```
#Two Dimensional Array 3 row, 3 column
b = [None]*3
b[0] = [1]*3
b[1] = [2]*3
b[2] = [3]*3
b[0][0], b[0][1], b[0][2] = 10,20,30
b[1][0], b[1][1], b[1][2] = 11,21,31
b[2][0], b[2][1], b[2][2] = 12,22,32
print  b
```

Output

```
>>>[[10, 20, 30], [11, 21, 31], [12, 22, 32]]
```

Lists and Loops

Listல் உள்ள valueகளை ஒவொன்றாக access செய்வதை for loop எளிமையாக்குகிறது. For loopன் syntaxல் listன் variable nameஐ கொடுக்கும்பொழுது அந்த loop listல் உள்ள valueகள் அனைத்தையும் access செய்து முடிக்கும் வரை executeஆகிறது.

Example

```
lsta = [1,2,3,4,5]
for x in lsta:
        print x,
```

Output

```
>>> 1 2 3 4 5
```

Assignment and Reference

Pythonல் assignement (=) என்பது referenceஐ copyசெய்யும் valueவை அல்ல.

Example

```
a = [1,2,3]
b = a
b[1] = 7
print "Value in a: ",a
```

Output

```
>>>Value in a: [1,7,3]
```

அதுவே reference இல்லாமல் valueஐ copy செய்வதற்கு இந்த syntax பயன்படுகிறது.

Syntax

```
list2 = list1[:]
```

Example

```
a = [1,2,3]
b = a[:]
b[2]=5
print "Value in A: ",a
print "Value in B: ",b
```

Output

```
>>>
Value in A:  [1, 2, 3]
Value in B:  [1, 2, 5]
```

Identity and Equality

Identityஐ check செய்வதற்கு is operatorம், equalityஐ check செய்வதற்கு double equal to (==) operatorம் உபயோகிக்கப்படுகிறது. Identityக்கும் equalityக்கும் உள்ள வித்தியாசம் என்னவென்றால் identity இரண்டு listகளை compare செய்யும்பொழுது இரண்டும் ஒரே objectஆ என்பதை சரிபார்க்கும் அதுவே equality இரண்டு listகளை compare செய்யும்பொழுது இரண்டின் அளவு மற்றும் அதில் உள்ள value ஒன்றாக இருக்கிறதா என்பதை சரிபார்க்கும்.

Example 1

```
#Equality
lst1 = [1,2,3]
lst2 = [1,2,3]
if lst1 == lst2:
        print "both are equal"
```

Example 2

```
#Identity
lst1 = [10,20,30]
```

```
lst2 = [10,20,30]
if lst1 is lst2:
        print "both are identical"
else:
        print "not identical"
```

Sorted List

Sorted listல் உள்ள itemகள் அனைத்தும் வரிசையாக இருக்கும். ஒரு listஐ sort செய்வதற்கு sorted function அல்லது sort method உபயோகிக்கப்படுகிறது.

Example

```
#sorted List
lst = [23,19,12,30,15,1,6,2]
print "Unsorted List", lst
lst = sorted(lst)
print "Sorted Function: ",lst
lst.append(2)
print "New List: ",lst
lst.sort()
print "Sorted Method: ", lst
```

Output

```
Unsorted List [23, 19, 12, 30, 15, 1, 6, 2]
Sorted Function:  [1, 2, 6, 12, 15, 19, 23, 30]
New List:  [1, 2, 6, 12, 15, 19, 23, 30, 2]
Sorted Method:  [1, 2, 2, 6, 12, 15, 19, 23, 30]
```

Tuples

Tuple என்பது listஐ போன்றது ஆனால் இதில் parenthesis (( )) உபயோகிக்கப்படுகிறது. Tupleல் valueவை ஒரு முறை assign செய்தபிறகு மாற்றமுடியாது. ஒரு tupleஐ listஆகவோ அல்லது listஐ tupleஆகவோ மாற்றமுடியும்.

Example

```
#tuple
tupa = (10,20,30,40)
print tupa
print tupa[3]
lsta = [12,30,15]
tupb = tuple(lsta)
print tupb
tupa[0]=20
```

Output

```
(10, 20, 30, 40)
40
(12, 30, 15)
Traceback (most recent call last):
  File "C:\Python\test.py", line 9, in <module>
        tupa[0]=20
TypeError: 'tuple' object does not support item assignment
```

இந்த example scriptல் tupa[0]=20 என்று valueவை assign செய்ய முயலும்பொழுது error வரும்.

Tuple and string formatting

Tupleஐ print statementல் valueகளை formattingஉடன் display செய்வதற்கு பயன்படுத்தப்படுகிறது. C programல் உள்ளதுபோல இதில் format specifier பயன்படுத்தப்படுகிறது.

Format Specifier	Data Type
%d	Integer Value

%s	String Value
%f	Float Value

Syntax

print “text with string formatting” %tuple
print “text with string formatting” %(values)

Example

```
#String Formatting
a = 10
b = 10.23
str1 = "welcome"
tup_fmt = (a,b,str1)
print  "Integer: %d\nFloat: %f\nString: %s" %tup_fmt
print "float %.1f\n Integer:%d" %(2.37,10)
```

String Functions

Python languageல் string valueகளை manipulate செய்வதற்கு இந்த functionகள் உபயோகிகப்படுகிறது.

Function	Description
s.capitalize()	Capitalizes first character of s
s.count(sub)	Count number of occurrences of sub in s
s.find(sub)	Find first index of sub in s, or -1 if not found
s.index(sub)	Find first index of sub in s, or raise ValueError if not found
s.rfind(sub)	Same as find, but last index
s.rindex(sub)	Same as index, but last index
s.lower()	Convert s to lower case
s.split()	Return a list of words in s
s.join(lst)	Join a list of words into a single string with s as separator
s.strip()	Strip leading/trailing white space from s
s.upper()	Convert s to upper string
s.replace(old, new)	Replace all instances of old with new in string

4.4 Dictionary - Introduction

Dictionaryஐ உருவாக்க braces ({ }) உபயோகிக்கப்படுகிறது. இதில் உள்ள valueவை access செய்வதற்கு indexஆக numberக்கு பதிலாக key பயன்படுத்தப்படுகிறது.

Syntax

dct = {Key:Value}

Example

```
dicta = {101:'arun',102:'balu',103:'mani'}
print dicta
        print dicta[101]
```

Output

```
>>> {101: 'arun', 102: 'balu', 103: 'mani'}
    arun
```

Dictionary Methods

Dictionaryல் கீழே கொடுக்கப்பட்டுள்ள methodsஐ பயன்படுத்தலாம்.

Method	Description
len(d)	Number of elements in d
d[k]	Item in d with key k
d[k] = v	Set item in d with key k to v
d.items()	Return a list of (key, value) pairs
d.keys()	Return a list of keys in d
d.values()	Return a list of values in d
d.clear()	Remove all items from dictionary d
d.copy()	Make a shallow copy of d

Combining two Dictionary

இரண்டு dictionaryல் உள்ள valueகளை இணைப்பதற்கு update method பயன்படுத்தப்படுகிறது. இதில் முதல் dictionary மற்றும் இரண்டாவது dictionaryல் ஒரே key இருந்தால் அது இரண்டு dictionaryயையம் இணைக்கும் பொழுது முதல் dictionaryல் உள்ள value அழிக்கப்பட்டு இரண்டாவது dictionaryல் உள்ள value storeஆகும்.

Example

```
dicta = {101:'arun',102:'balu',103:'mani'}
dictb = {101:'anbu',104:'ramu',105:'rajesh'}
dicta.update(dictb)
print dicta
```

Output

```
>>>{101: 'anbu', 102: 'balu', 103: 'mani', 104: 'ramu', 105: 'rajesh'}
```

Making copies

Pythonல் assignment (=) மூலமாக ஒரு dictionaryஐ மற்றொரு dictionaryக்கு assign செய்யும் பொழுது அது referenceஐ மட்டுமே எடுத்துக்கொள்ளும் valueவை அல்ல, அதனால் ஒரு dictionaryஐ மற்றொரு dictionaryக்கு copy செய்ய copy() method பயன்படுத்தப்படுகிறது.

Example

```
dicta = {101:'arun',102:'balu',103:'mani'}
dictb = dicta
dictb[101]='Anbu'
print "Dict A: ",dicta
dictc = dicta.copy()
dictc[101]='Antony'
print "Dict A: ",dicta
print "Dict C: ",dictc
```

Output

```
Dict A:  {101: 'Anbu', 102: 'balu', 103: 'mani'}
Dict A:  {101: 'Anbu', 102: 'balu', 103: 'mani'}
Dict C:  {101: 'Antony', 102: 'balu', 103: 'mani'}
```

Persistent Variables

Persistent Variable என்ற variable அதன் valueவை பல program executeஆகும் பொழுது அனைத்திலும் அதன் value கிடைக்கப்பெறும்மாரு இருக்கும். இதற்கு pythonல் shelve என்கிற module உபயோகிக்கப்படுகிறது. இதன் மூலம் ஒரு valueவை பல programல் access செய்யுமாறு store செய்யலாம்.

Example store.py

```
#Persistent Storage
```

```
import shelve
data = shelve.open("info")
data["perval"] = 20
print "Persistent Value: ",data["perval"]
data.close()
```

Output

```
Persistent Value:  20
```

Example get.py

```
#Persistent Storage
import shelve
data = shelve.open("info")
print "Accessing Persistent Value across Files"
print "Persistent Value: ",data["perval"]
```

Output

```
Accessing Persistent Value across Files
Persistent Value:  20
```

முதல் program store.pyல் நாம் shelveவின் மூலமாக store செய்யும் info enkira variable இரண்டாவது programஆன get.pyல் நம்மால் access செய்ய முடியும்.

Internal Dictionaries

Pythonல் ஒரு program executeஆகும் பொழுது அதில் உள்ள variable மற்றும் அதற்கான valueகளை store செய்வதற்கு dictionaryஐ பயன்படுத்துகிறது. இந்த dictionaryஐ access செய்வதற்கு locals() function பயன்படுகிறது.

Example

```
#locals
a = 10
b = 20
print locals()
c = a+b
print locals()
```

Output

```
{'a': 10, 'b': 20}
{'a': 10, 'c': 30, 'b': 20}
```

4.5 Functions and Files

Functions

Function என்பது ஒரு குறிப்பிட்ட பணியை செய்வதற்கான statementகளின் தொகுப்பாகும் இதற்கு ஒரு பெயர் குறிப்பிடப்பட்டிருக்கும். Pythonல் functionகளை உருவாக்க def keyword பயன்படுகிறது. மற்ற languageகளில் இருப்பதுபோல python languageல் function statementகளை braces ({ }) வரையருக்காது மாறாக ஒரு function statementகளின் முடிவை return statement வரையறுக்கும்.

Syntax

```
def function_name(parameter)
        statements
return
```

Example

```
def add():
  a = input("Enter A Value: ")
  b = input("Enter B Value: ")
  res = a + b
  print "Sum is ",res
  return

#main Script
add()
```

Output

```
Enter A Value: 10
Enter B Value: 20
Sum is  30
```

Python function argumentsஐ பொறுத்து நன்கு வகைப்படும்

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length arguments

Required arguments

Required arguments என்பதில் functionல் எதனை parameterகள் உள்ளதோ, அனைத்தையும் function callல் கொடுக்கவேண்டும் இல்லையென்றால் programல் error வரும்.

Example

```
def add(a,b):
  res = a + b
  print "Sum is ",res
  return
#main Script
add(10,20)
add (10)
```

Output

```
Sum is  30
Traceback (most recent call last):
  File "C:\Python\24-fun-req.py", line 8, in <module>
add (10)
TypeError: add() takes exactly 2 arguments (1 given)
```

Keyword Arguments

Keyword Arguments என்பதில் functionல் உள்ள parameterகளுக்கு அதன் பெயரைகொண்டு value அனுப்பப்படுகிறது. இதனால் pythonல் parameterஐ எந்த வரிசையில் வேண்டுமானாலும் கொடுக்கலாம்.

Example

```
def add(a,b):
res = a + b
  print "a(%d)+b(%d)=%d " %(a,b,res)
  return
#main Script
add(a=10,b=20)
add(b=15,a=4)
```

Output

```
a(10)+b(20)=30
a(4)+b(15)=19
```

Default arguments

Default arguments என்பதில் ஒரு functionஐ call செய்யும் பொழுது அதில் உள்ள parameterக்கு value கொடுக்கவில்லை என்றல் functionஐ define செய்யும்பொழுது parameterக்கு கொடுக்கப்பட்ட valueவை defaultஆக எடுத்துக்கொள்ளும்.

Example

```
def add(a,b=10):
  res = a + b
  print "a(%d)+b(%d)=%d " %(a,b,res)
  return

#main Script
add(5,23)
add(4)
```

Output

```
a(5)+b(23)=28
a(4)+b(10)=14
```

இதில் முதல் function callலில் இரண்டு parameterகளும் கொடுக்கப்பட்டுள்ளது. இரண்டாவது function callலில் முதல் parameterஆனா aக்கு மட்டும் value கொடுக்கப்பட்டுள்ளது parameter bக்கு defaultஆக கொடுக்கப்பட்டுள்ள 10 என்கிற valueஐ python தானாக எடுத்துக்கொள்ளும்.

Variable-length arguments

Variable-length arguments என்பது ஒரு funcitonல் ஒன்றுக்கும் மேற்பட்ட parameterகளை program executeஆகும் பொழுது பெறுவதற்கு பயன்படுவதாகும்.

Example

```
def add(arg,*varlen):
   res = 0
   res = res + arg
   for x in varlen:
      res = res + x
   print "Sum of %d numbers is %d " %(len(varlen)+1,res)


#main Script
add(5,23)
add(4,1,2,3,4)
```

Output

```
Sum of 2 numbers is 28
Sum of 5 numbers is 14
```

File Handling

Python languageல் file operations செய்வதற்கு முதலில் open() functionஐ கொண்டு file object உருவாக்கப்படுகிறது இதன் உதவியுடன் மற்ற file operationகள் செய்யப்படுகிறது. Pythonல் ஒரு fileலில் நான்கு operationகள் செய்யமுடியும் அவை

- Opening a File
- Closing a File
- Writing a File
- Reading a File

Opening a File

ஒரு fileஐ open செய்வதற்கு open() function பயன்படுத்தப்படுகிறது. இதில் இரண்டு parameterகள் உள்ளன, அவை file name மற்றும் file opening mode ஆகும்.

Syntax

```
file_obj = open(filename [,mode])
```

File Opening Modes

- r - reading
- w - writing
- a - append
- b - binary
- + - read and write

Closing a File

Open செய்யப்பட்ட fileஐ close செய்வதற்கு close() function பயன்படுகிறது. ஒரு fileஐ close செய்தபிறகு அதில் வேறு data எதுவும் எழுதுவது தவிர்க்கப்படுகிறது.

Syntax

file_obj.close()

Writing a File

Open செய்யப்பட்ட fileல் text dataவை எழுதுவதற்கு write() function பயன்படுகிறது.

Syntax

file_obj.write(content)

Reading a File

ஒரு fileல் உள்ள text dataவை படிப்பதற்கு read() funtion அல்லது file object பயன்படுத்தப்படுகிறது. read() function fileல் உள்ள மொத்த contentஐ read செய்கிறது. File object for loopன் உதவியுடன் fileல் உள்ள ஒருவரு வாரியாக read செய்கிறது.

Syntax

file_obj.read()

Example Script

```
fo = open("sample.txt","w")
text = "Writing into file"
fo.write(text)
text = "\nSome Content"
fo.write(text)
fo.close()
print "File written"
fo = open("sample.txt","r")
text = fo.read()
print text
fo.close()
fo = open("sample.txt","r")
print "Reading File Using File Object"
for line in fo:
   print "Line: ",line,
fo.close()
```

Output

```
File written
Writing into file
Some Content
Reading File Using File Object
Line:  Writing into file
Line:  Some Content
```

Exception Handling

Exception என்பது ஒரு script executeஆகும் பொழுது அதன் executionல் தடங்கல் வந்து அந்த script சரியான பாதையில் execute ஆகாமல் இருப்பதாகும். இதனை சரியான முறையில் கையாளவில்லை என்றல் script execution பத்தியில் நின்றுவிடும் Error வந்தாலும் script எந்ததொரு தடங்கலும் இல்லாமல் அதன் executionஐ முடிப்பதற்கு exception handling பயன்படுகிறது. Error வரக்கூடிய codeஐ try: blockலும், அந்த errorஐ கையாளக்கூடிய codeஐ except: blockலும் கொடுக்கின்றோம்.

Syntax

```
try:
   # Code that Generate Exception
except:
   # Handle Exception
```

Pythonல் பொதுவாக வரக்கூடிய exceptionகள்

- Exception - General Error Class
- IOError - Input Output Error
- ImportError - Library Not Found Error
- ValueError - Empty Value Error
- ZeroDivisionError - Arithmetic Exception
- EOFError - File End Reached Error

Example

```
#simple Exception
```

```
try:
   fobj = open("test.txt","r")
   fobj.write("Welcome")
   fobj.close()
except IOError:
   print "Error in accessing file"
```

Output

Error in accessing file

Multiple Exceptions

ஒரு scriptல் ஒன்றுக்கும் மேற்பட்ட errorகள் வரலாம் என்றல் அந்த scriptஐ try: blockகிலும் அதற்காக பல except: blockகளும் பயன்படுத்தப்படுகிறது.

Example

```
#Multiple Exception
try:
   fobj = open("test.txt","r")
   fobj.write("Welcome")
   fobj.close()
   txt = fobj.read()
except IOError:
   print "Error in accessing file"
except ValueError:
   print "Closed file cannot be read"
```

Finally Block

ஒரு scriptல் error வந்தாலும் வரவில்லை என்றாலும் execute ஆகவேண்டிய statementகளை finally blockல் கொடுப்போம்.

Example

```
#Finally Clause
try:
   fobj = open("test.txt","r")
   fobj.write("Welcome")
   fobj.close()
   print "File Closed "
except IOError:
   print "Error in accessing file"
finally:
   print "Closing the File in finally"
   fobj.close()
```

User defined Exception

Defaultஆக python languageல் உள்ள exceptionகளை தவிர வேறு execptionகளை உருவாக்க user defined exception முறை பயன்படுகிறது.

Syntax

```
class UserError(Exception):
   def __init__(self,data):
      self.data = data

raise UserError('Error Text')
```

Example

```
class MarkError(Exception):
def __init__(self,data):
self.data = data

print "Enter 5 subject marks"
tot = 0
i = 0
for x in range(0,5):
   try:
         mark = input("Enter mark")
         if mark < 0 or mark > 100:
            raise MarkError("Enter Marks 0-100")
         else:
            tot = tot + mark
            i = i +1
   except MarkError as e:
```

```
        print e.data
print "Total of %d subjects is %d" %(i,tot)
```

இந்த Scriptல் mark value userரிடம் இருந்து வாங்கும்பொழுது அதன் மதிப்பு 0வை விட குறைவாகவோ அல்லது 100க்கு அதிகமாகவோ இருந்தால் MarkError என்கிற user defined exceptionன் உருவாக்கப்படுகிறது.

Output

```
Enter 5 subject marks
Enter mark78
Enter mark90
Enter mark150
Enter Marks 0-100
Enter mark86
Enter mark67
Total of 4 subjects is 321
```

.