**# 6\_Basic Instructions for Python**

**Created by : Yogendra Patil**

**PhD, University of Alabama**

Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> print "Hello All!"

Hello All!

>>> 2+2

4

>>> 6 - 3

3

>>> 18/7

2

>>> 18/7.0

2.5714285714285716

>>> 18/7.

2.5714285714285716

>>> 9%4

1

>>> 8%4

0

>>> 8.75%.5

0.25

>>> 6\*7

42

>>> 8^3

11

>>> 8\*\*3

512

>>> 5\*\*5

3125

>>> -7\*\*2

-49

>>> x = 18

>>> x + 15

33

>>> y = 54

>>> x + y

72

>>> x+y

72

>>> g = input("Enter number here :")

Enter number here :45

>>> g + 2

47

>>> clc

Traceback (most recent call last):

File "<pyshell#21>", line 1, in <module>

clc

NameError: name 'clc' is not defined

>>> clear

Traceback (most recent call last):

File "<pyshell#22>", line 1, in <module>

clear

NameError: name 'clear' is not defined

>>> pow(5,2)

25

>>> abs(-4)

4

>>> floor(18.6)

Traceback (most recent call last):

File "<pyshell#25>", line 1, in <module>

floor(18.6)

NameError: name 'floor' is not defined

>>> import math

>>> floor(18.9)

Traceback (most recent call last):

File "<pyshell#27>", line 1, in <module>

floor(18.9)

NameError: name 'floor' is not defined

>>> math.floor(17.5)

17.0

>>> math.sqrt(4)

2.0

>>> bucky = math.sqrt

>>> bucky(9)

3.0

>>> ================================ RESTART ================================

>>>

Hello all!!

>>> ================================ RESTART ================================

>>>

Enter name:yogi

Heyyogi

>>> ================================ RESTART ================================

>>>

Enter name:yogi

Hey yogi

>>> ================================ RESTART ================================

>>>

Enter name:yogi

Hey yogi

Press<enter>

>>> ================================ RESTART ================================

>>>

Enter name:yogi

Traceback (most recent call last):

File "C:/Python27/FirstCode.py", line 1, in <module>

x = input("Enter name:")

File "<string>", line 1, in <module>

NameError: name 'yogi' is not defined

>>> ================================ RESTART ================================

>>>

Enter name:u

Traceback (most recent call last):

File "C:/Python27/FirstCode.py", line 1, in <module>

x = input("Enter name:")

File "<string>", line 1, in <module>

NameError: name 'u' is not defined

>>> ================================ RESTART ================================

>>>

Enter name:i

Hey i

Press<enter>

>>> ================================ RESTART ================================

>>> "Hey Now"

'Hey Now'

>>> 'Hey now'

'Hey now'

>>> 'he\'s a jerk'

"he's a jerk"

>>> "yogi says,\"How are you\""

'yogi says,"How are you"'

>>> a = 'rock'

>>> b = 'buck'

>>> a + b

'rockbuck'

>>> a \* b

Traceback (most recent call last):

File "<pyshell#39>", line 1, in <module>

a \* b

TypeError: can't multiply sequence by non-int of type 'str'

>>> num = 18

>>> num + 16

34

>>> print "Yogi is " + num

Traceback (most recent call last):

File "<pyshell#42>", line 1, in <module>

print "Yogi is " + num

TypeError: cannot concatenate 'str' and 'int' objects

>>> print "Yogi is " + str(num)

Yogi is 18

>>> print "Yogi is " + 'num'

Yogi is num

>>> num = 30

>>> print "Yogi is " + `num`

Yogi is 30

>>> ================================ RESTART ================================

>>>

Enter name:o

Traceback (most recent call last):

File "C:/Python27/FirstCode.py", line 4, in <module>

yogi = input("Enter name:")

File "<string>", line 1, in <module>

NameError: name 'o' is not defined

>>> ================================ RESTART ================================

>>>

Enter name:you

>>> buck

Traceback (most recent call last):

File "<pyshell#47>", line 1, in <module>

buck

NameError: name 'buck' is not defined

>>> yogi

'you'

>>> ================================ RESTART ================================

>>>

Enter name:yogi

yogi

>>> family = ['mom','dad','bro','sis','dog']

>>> family

['mom', 'dad', 'bro', 'sis', 'dog']

>>> family(1)

Traceback (most recent call last):

File "<pyshell#51>", line 1, in <module>

family(1)

TypeError: 'list' object is not callable

>>> family[1]

'dad'

>>> family[-3]

'bro'

>>> family[-1]

'dog'

>>> family[0]

'mom'

>>> 'bucky'[3]

'k'

>>> g = 'yogi'

>>> g[0]

'y'

>>> example = [0,1,2,3,4,5,6,7,8,9]

>>> example[4:8]

[4, 5, 6, 7]

>>> example[4:9]

[4, 5, 6, 7, 8]

>>> example[4:10]

[4, 5, 6, 7, 8, 9]

>>> example[4:11]

[4, 5, 6, 7, 8, 9]

>>> example[4:15]

[4, 5, 6, 7, 8, 9]

>>> example[-5:0]

[]

>>> example[-1:-1]

[]

>>> example[-1:-5]

[]

>>> example[-5:]

[5, 6, 7, 8, 9]

>>> example[3:]

[3, 4, 5, 6, 7, 8, 9]

>>> example[:3]

[0, 1, 2]

>>> example

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> example[1:8:2]

[1, 3, 5, 7]

>>> example[8:0:2]

[]

>>> example[-3:0:2]

[]

>>> example[2:0:-3]

[2]

>>> example

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> example[4:8]

[4, 5, 6, 7]

>>> example[:-5]

[0, 1, 2, 3, 4]

>>> example[5:]

[5, 6, 7, 8, 9]

>>> example[1:5:2]

[1, 3]

>>> example[0::2]

[0, 2, 4, 6, 8]

>>> example[0:8:2]

[0, 2, 4, 6]

>>> example[10:0:-2]

[9, 7, 5, 3, 1]

>>> example[9::-2]

[9, 7, 5, 3, 1]

>>> example[11::-1]

[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

>>> [7,4,5]+[4,5,6]

[7, 4, 5, 4, 5, 6]

>>> 'bucky'+'rocky'

'buckyrocky'

>>> numbers = [8,1,14,17,16]

>>> len(numbers)

5

>>> list('yogi ')

['y', 'o', 'g', 'i', ' ']

>>> numbers[5] = 100

Traceback (most recent call last):

File "<pyshell#91>", line 1, in <module>

numbers[5] = 100

IndexError: list assignment index out of range

>>> numbers(5) = 100

SyntaxError: can't assign to function call

>>> numbers[3] = 100

>>> numbers

[8, 1, 14, 100, 16]

>>> del numbers[3]

>>> numbers

[8, 1, 14, 16]

>>> example = list('easyhoss')

>>> example

['e', 'a', 's', 'y', 'h', 'o', 's', 's']

>>> example[4:] = list('boss')

>>> example

['e', 'a', 's', 'y', 'b', 'o', 's', 's']

>>> example[4:] = list('race casr')

>>> example

['e', 'a', 's', 'y', 'r', 'a', 'c', 'e', ' ', 'c', 'a', 's', 'r']

>>> example = [7,8,9]

>>> example

[7, 8, 9]

>>> example[1:1] = [3,3,3]

>>> example

[7, 3, 3, 3, 8, 9]

>>> example[1:2] = [5,5,5]

>>> example

[7, 5, 5, 5, 3, 3, 8, 9]

>>> example[1:5] = []

>>> example

[7, 3, 8, 9]

>>> face = [21,18,30]

>>> face

[21, 18, 30]

>>> face.append

<built-in method append of list object at 0x02C3FF30>

>>> face.append(45)

>>> face

[21, 18, 30, 45]

>>> apples = ['i','love','apples','apples','now']

>>> apples

['i', 'love', 'apples', 'apples', 'now']

>>> apples.count('apples')

2

>>> apples.count('you')

0

>>> apples.append('bow')

>>> apples

['i', 'love', 'apples', 'apples', 'now', 'bow']

>>> one = [1,2,3]

>>> two = [4,5,5]

>>> one.extend(two)

>>> one

[1, 2, 3, 4, 5, 5]

>>> one+two

[1, 2, 3, 4, 5, 5, 4, 5, 5]

>>> say = ['hey','now','brown','cow']

>>> say.index('brown')

2

>>> say.insert(2,'hoss')

>>> say

['hey', 'now', 'hoss', 'brown', 'cow']

>>> say.pop(1)

'now'

>>> say

['hey', 'hoss', 'brown', 'cow']

>>> say(1)

Traceback (most recent call last):

File "<pyshell#133>", line 1, in <module>

say(1)

TypeError: 'list' object is not callable

>>> say.remove('brown')

>>> say

['hey', 'hoss', 'cow']

>>> say.count('hoss')

1

>>> new = [32,54,22,7,98,1]

>>> new

[32, 54, 22, 7, 98, 1]

>>> new.sort()

>>> new

[1, 7, 22, 32, 54, 98]

>>> sorted('Easyhoss')

['E', 'a', 'h', 'o', 's', 's', 's', 'y']

>>> 41,42,32,54

(41, 42, 32, 54)

>>> bucky = (32,33,43,54)

>>> bucky

(32, 33, 43, 54)

>>> bucky[2]

43

Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> seq = ['hey' ,'there', 'bessie', 'hoss']

>>> seq

['hey', 'there', 'bessie', 'hoss']

>>> glue = 'moss'

>>> glue.join(seq)

'heymosstheremossbessiemosshoss'

>>> randstr = 'I wish I ahd no sausage'

>>> randstr.lower()

'i wish i ahd no sausage'

>>> randstr.upper()

'I WISH I AHD NO SAUSAGE'

>>> truth = "I love old women"

>>> truth.replace('women','girls')

'I love old girls'

>>> book = {'Dad':'Bob','Mom':'Lisa','Bro':'Joe'}

>>> book

{'Dad': 'Bob', 'Bro': 'Joe', 'Mom': 'Lisa'}

>>> book['Dad']

'Bob'

>>> book[Mom]

Traceback (most recent call last):

File "<pyshell#12>", line 1, in <module>

book[Mom]

NameError: name 'Mom' is not defined

>>> book['Mom']

'Lisa'

>>> ages = {'Dad':'42','Mom':'87','Bro':'23'}

>>> ages

{'Dad': '42', 'Bro': '23', 'Mom': '87'}

>>> # keys and value e.g. {Key:Value,Key:Value}

>>> ages['Dad']

'42'

>>> book.clear()

>>> book

{}

>>> tuna= ages.copy()

>>> tuna

{'Dad': '42', 'Bro': '23', 'Mom': '87'}

>>> tuna.has\_key("Mom")

True

>>> tuna.has\_key("Sis")

False

>>> ================================ RESTART ================================

>>>

this is a fish alright

>>> ================================ RESTART ================================

>>>

>>> ================================ RESTART ================================

>>>

>>> ================================ RESTART ================================

>>>

I dont know what is it

>>> ================================ RESTART ================================

>>>

This is Tuna

>>> ================================ RESTART ================================

>>>

This is Tuna

>>> ================================ RESTART ================================

>>>

I dont know what is it

>>> ================================ RESTART ================================

>>>

this is cat

>>> 9 > 9

False

>>> 9 <= 9

True

>>> 9 != 3

True

>>> 9 != 9

False

>>> one = [21,22,23]

>>> two = [21,22,23]

>>> one == two

True

>>> one is two

False

>>> three = four = [2,3,4]

>>> three is four

True

>>> pizza = 'pizzahut'

>>> s in pizza

Traceback (most recent call last):

File "<pyshell#35>", line 1, in <module>

s in pizza

NameError: name 's' is not defined

>>> 's' in pizza

False

>>> 'p' in pizza

True

>>> if 'p' in pizza:

print 'Yes'

Yes

>>> 'dog' < 'cat'

False

>>> example = 5

>>> if example > 3 and example < 10

SyntaxError: invalid syntax

>>> if example > 3 and example < 10:

print 'number is between 3 and 10'

number is between 3 and 10

>>> if example > 3 and example < 4:

print 'x'

>>> if example > 3 or example < 10:

print 'Yes'

Yes

>>> if example > 3 or example < 10:

print 'hey now'

hey now

>>> b = 1

>>> b

1

>>> while b <= 10:

print b

b += 1

1

2

3

4

5

6

7

8

9

10

>>> gl = ['bread','milk','meat','beef']

>>> gl

['bread', 'milk', 'meat', 'beef']

>>> for food in gl:

print 'I want ' + food

I want bread

I want milk

I want meat

I want beef

>>> ages = {'dad': 42,'mom':48,'lisa':7}

>>> ages

{'dad': 42, 'lisa': 7, 'mom': 48}

>>> for item in ages:

print item

dad

lisa

mom

>>> for item in ages:

print item, ages[item]

dad 42

lisa 7

mom 48

>>> while 1:

name = raw\_input('Enter name: ' )

if name == 'quit':break

Enter name: craig

Enter name: tom

Enter name: lisa

Enter name: quit

>>> def whatsup(x):

return 'whats up' + x

>>> print whatsup('ony')

whats upony

>>> print whatsup('Loop')

whats upLoop

>>> def plusten(y):

return y + 10

>>> print plusten(45)

55

>>> def name(first,last):

print '%s %s' % (first,last)

>>> name('Yogi ','Bearr')

Yogi Bearr

>>> def name(first='tom',last = 'smith'):

print '%s %s' % (first,last)

>>> name()

tom smith

>>> name('Yogi ','Roberts')

Yogi Roberts

>>> naem()

Traceback (most recent call last):

File "<pyshell#98>", line 1, in <module>

naem()

NameError: name 'naem' is not defined

>>> name()

tom smith

>>> name('Yogi')

Yogi smith

>>> name(last = 'Bearr')

tom Bearr

>>> def list(\*food):

print food

>>> list('apples')

('apples',)

>>> list('apples','peaches','beef')

('apples', 'peaches', 'beef')

>>> def profile(name,\*ages):

print name

print ages

>>> profile('Yogi',42,43,76,98,54)

Yogi

(42, 43, 76, 98, 54)

>>> # Tuple

>>> def cart(\*\*items):

print items

>>> cart(apples = 4,peaches = 6,beef = 60)

{'peaches': 6, 'apples': 4, 'beef': 60}

>>> # Dictionary = Items : Values

>>> def profile(first,last,\*ages,\*\*items):

print first,last

print ages

print items

>>> profile('yogi','bearr',32,33,23,45,bacon = 4,sausage = 45)# Tuples and Dictionary

yogi bearr

(32, 33, 23, 45)

{'sausage': 45, 'bacon': 4}

>>> def example(a,b,c):

return a + b \* c

>>> tuna = (5,7,3) # Tuple not Dixtionary

>>> example(\*tuna) # \* - means its a tuple (array)

26

>>> def example2(\*\* this):

print this

>>> bacon = {'Mom':32,'dad':54} # This is a Dictionary

>>> example2(\*\*bacon) # \* - means its a Dictionary

{'dad': 54, 'Mom': 32}

>>>

Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> # 32 Tutorial

>>> # Objects Oriented Programming

>>> # Object.Attribute (to describe objects)

>>> # Object.Method()

>>> # But you first need to build a class

>>> # to perform a method

>>> class exampleClass:

eyes = "blue"

age = 22

def thisMethod(self):# Method (self)

return 'Hery this Method works'

>>> exampleClass

<class \_\_main\_\_.exampleClass at 0x0237CB58>

>>> # This is how build a class

>>> exampleObject = exampleClass()# create a Objects that is blueprint of the class, so as to implememnt the methods inside the classd

>>> exampleObject.eyes

'blue'

>>> # You accessed the value of the var 'eyes' in the class

>>> exampleObject.thisMethod()

'Hery this Method works'

>>> exampleObject.age

22

>>> class className:

def createName(self,name):

self.name = name

def displayName(self):

return self.name

def saying(self):

print 'Hello %s' % self.name

>>> className

<class \_\_main\_\_.className at 0x029E6ED8>

>>> # to check if your class was created

>>> # In order to access elements in the class you need to First Create and Object

>>> first = className()# this is how you create a n Object

>>> second = className()

>>> first.createName('bucky')

>>> second.createName('Tony')

>>> first.displayName()

'bucky'

>>> first.saying()

Hello bucky

>>> # make sure all the def's are in one line

>>> first.name

'bucky'

>>> # 34 Tut Subclasses Superclasses

>>> class parentClass:

var1 = 'I am var1'

var2 = 'I am var2'

>>> class childClass(parentClass):# to inherent the class

pass

>>> # pass means dont do any thing, just pass

>>> # to inherent main class(parent) genes

>>> parentObject = parentClass()

>>> parentObject.var1

'I am var1'

>>> childObject = childClass()

>>> childObject.var1

'I am var1'

>>> childObject.var2

'I am var2'

>>> # Overwrite Var in subclasses 35 Tut

>>> class parent:

var1 = 'bacon'

var2 = 'sausage'

>>> class child(parent):# subclass(superclass)

var2 = 'toast' # overwrite var2 = toast instead of sausage

>>> pob = parent()

>>> cob = child()

>>> pob.var1

'bacon'

>>> pob.var2

'sausage'

>>> cob.var1

'bacon'

>>> cob.var2

'toast'

>>> # 36 Tut Multiple Parent Clas

>>> class Mom:

var1 = 'Hey Im mom'

>>> class dad:

var2 = 'Hey there sonny Im dad'

>>> class child(Mom,dad):

var3 = 'im a neew var'

>>> childObject = child()

>>> childObject.var1

'Hey Im mom'

>>> # child inhereted elements of Main/Parent class 'Mom','dad'

>>> childObject.var2

'Hey there sonny Im dad'

>>> childObject.var3

'im a neew var'

>>> # Tut 37 Constructurs

>>> class swine:

def apples(self):

print "beef pie"

>>> obj = swine()

>>> obj.apples()

beef pie

>>> class new:

def \_\_init\_\_(self): # Two underscores in-built routinr

print 'this is a constructuree'

print 'this also print out'

>>>

>>> newobj = new()

this is a constructuree

this also print out

>>> # So you dont have to call the methods using objects

>>> # It automatically executes the initialisation part

>>> # Tut 38 Import Modules

>>> import swineflu # you created in the editor

>>> # you just imported all modeules in teh swine flu

>>> swineflu.testMod

<function testMod at 0x02B1EC70>

>>> swineflu.testMod()

this baby worked

>>> import swineflu

>>> swineflu.testMod()

this baby worked

>>> # you need to close this shell and open it again

>>> # to get this baby is fat

>>>

>>> # Tut 39 Reload Modules

>>> import swineflu

>>> swineflu.testMod()

this baby worked

>>> baby = swineflu.testMod()

this baby worked

>>> baby()

Traceback (most recent call last):

File "<pyshell#112>", line 1, in <module>

baby()

TypeError: 'NoneType' object is not callable

>>> baby = swineflu.testMod

>>> baby

<function testMod at 0x02B1EC70>

>>> baby()

this baby worked

>>> # you dont need to close shell

>>> reload(swineflu)

<module 'swineflu' from 'swineflu.py'>

>>> swineflu.testMod

<function testMod at 0x02B20BB0>

>>> swineflu.testMod()

this baby is fat

>>>

>>> # 40 Tut Getting Module

>>> import math

>>> math.sqrt(81)

9.0

>>> dir(math)

['\_\_doc\_\_', '\_\_name\_\_', '\_\_package\_\_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

>>> help(math)

Help on built-in module math:

NAME

math

FILE

(built-in)

DESCRIPTION

This module is always available. It provides access to the

mathematical functions defined by the C standard.

FUNCTIONS

acos(...)

acos(x)

Return the arc cosine (measured in radians) of x.

acosh(...)

acosh(x)

Return the hyperbolic arc cosine (measured in radians) of x.

asin(...)

asin(x)

Return the arc sine (measured in radians) of x.

asinh(...)

asinh(x)

Return the hyperbolic arc sine (measured in radians) of x.

atan(...)

atan(x)

Return the arc tangent (measured in radians) of x.

atan2(...)

atan2(y, x)

Return the arc tangent (measured in radians) of y/x.

Unlike atan(y/x), the signs of both x and y are considered.

atanh(...)

atanh(x)

Return the hyperbolic arc tangent (measured in radians) of x.

ceil(...)

ceil(x)

Return the ceiling of x as a float.

This is the smallest integral value >= x.

copysign(...)

copysign(x, y)

Return x with the sign of y.

cos(...)

cos(x)

Return the cosine of x (measured in radians).

cosh(...)

cosh(x)

Return the hyperbolic cosine of x.

degrees(...)

degrees(x)

Convert angle x from radians to degrees.

erf(...)

erf(x)

Error function at x.

erfc(...)

erfc(x)

Complementary error function at x.

exp(...)

exp(x)

Return e raised to the power of x.

expm1(...)

expm1(x)

Return exp(x)-1.

This function avoids the loss of precision involved in the direct evaluation of exp(x)-1 for small x.

fabs(...)

fabs(x)

Return the absolute value of the float x.

factorial(...)

factorial(x) -> Integral

Find x!. Raise a ValueError if x is negative or non-integral.

floor(...)

floor(x)

Return the floor of x as a float.

This is the largest integral value <= x.

fmod(...)

fmod(x, y)

Return fmod(x, y), according to platform C. x % y may differ.

frexp(...)

frexp(x)

Return the mantissa and exponent of x, as pair (m, e).

m is a float and e is an int, such that x = m \* 2.\*\*e.

If x is 0, m and e are both 0. Else 0.5 <= abs(m) < 1.0.

fsum(...)

fsum(iterable)

Return an accurate floating point sum of values in the iterable.

Assumes IEEE-754 floating point arithmetic.

gamma(...)

gamma(x)

Gamma function at x.

hypot(...)

hypot(x, y)

Return the Euclidean distance, sqrt(x\*x + y\*y).

isinf(...)

isinf(x) -> bool

Check if float x is infinite (positive or negative).

isnan(...)

isnan(x) -> bool

Check if float x is not a number (NaN).

ldexp(...)

ldexp(x, i)

Return x \* (2\*\*i).

lgamma(...)

lgamma(x)

Natural logarithm of absolute value of Gamma function at x.

log(...)

log(x[, base])

Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.

log10(...)

log10(x)

Return the base 10 logarithm of x.

log1p(...)

log1p(x)

Return the natural logarithm of 1+x (base e).

The result is computed in a way which is accurate for x near zero.

modf(...)

modf(x)

Return the fractional and integer parts of x. Both results carry the sign

of x and are floats.

pow(...)

pow(x, y)

Return x\*\*y (x to the power of y).

radians(...)

radians(x)

Convert angle x from degrees to radians.

sin(...)

sin(x)

Return the sine of x (measured in radians).

sinh(...)

sinh(x)

Return the hyperbolic sine of x.

sqrt(...)

sqrt(x)

Return the square root of x.

tan(...)

tan(x)

Return the tangent of x (measured in radians).

tanh(...)

tanh(x)

Return the hyperbolic tangent of x.

trunc(...)

trunc(x:Real) -> Integral

Truncates x to the nearest Integral toward 0. Uses the \_\_trunc\_\_ magic method.

DATA

e = 2.718281828459045

pi = 3.141592653589793

>>> math.\_\_doc\_\_

'This module is always available. It provides access to the\nmathematical functions defined by the C standard.'

>>> dir(math)

['\_\_doc\_\_', '\_\_name\_\_', '\_\_package\_\_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

>>> # you dont use the one with two underscores

>>>

>>>

>>>

>>> # 41 Tut

>>> # how to open a text file

>>> fob = open('c:/testPy/a.txt','w')# w means write

>>> fob.write('Hey now brown cow')

>>> fob.close()

>>> fob = open('c:/testPy/a.txt','r')# r means read

>>> fob.read(3)

'Hey'

>>> fob.read()

' now brown cow'

>>> fob.read(:)

SyntaxError: invalid syntax

>>> fob.read()

''

>>> fob.read(1:)

SyntaxError: invalid syntax

>>> fob.close()

>>> # dont forget to close your files

>>>

>>> # 42 Tut read and write

>>> fob = open('c:/testPy/a.txt','r')# r means read

>>> print fob.readline()

Hey now brown cow

>>> print fob.readlines()

[]

>>> fob.close()

>>> fob = open('c:/testPy/a.txt','w')

>>> fob.write('this is a new line \n this is line 2\n this is 3rd line \n thi is final line')

>>> fob.close()

>>> print fob.readlines()

Traceback (most recent call last):

File "<pyshell#154>", line 1, in <module>

print fob.readlines()

ValueError: I/O operation on closed file

>>> fob = open('c:/testPy/a.txt','r')

>>> print fob.readlines()

['this is a new line \n', ' this is line 2\n', ' this is 3rd line \n', ' thi is final line']

>>> fob = open('c:/testPy/a.txt','r')

>>> listme = fob.readlines()

>>> listme

['this is a new line \n', ' this is line 2\n', ' this is 3rd line \n', ' thi is final line']

>>> fob.close()

>>> listme[2] # means third line not 2nd

' this is 3rd line \n'

>>> fob.close()

>>> listme[2] = 'mm im sure i love bacon'

>>> listme

['this is a new line \n', ' this is line 2\n', 'mm im sure i love bacon', ' thi is final line']

>>> fob = open('c:/testPy/a.txt','w')

>>> fob.writelines(listme)# there is no writeline func

>>> fob.close()

>>>

>>>

>>> # 44 Tut Installing Wx Python

>>>