

Lists

Create an empty list	<code>newlist=[]</code>
Assign value at index	<code>alist[index]= value</code>
Access value at index	<code>alist[index]</code>
Add item to list	<code>alist.append(new item)</code>
Insert into list	<code>alist.insert(at position, new item)</code>
Count # of an item in list	<code>alist.count(item)</code>
Delete 1 matching item	<code>alist.remove(del item)</code>
Remove item at index	<code>del alist[index]</code>

Looping

For loop 0 thru 9	<code>for x in range(10):</code>
For loop 5 thru 10	<code>for x in range(5,11):</code>
For each char in a string	<code>for char in astring:</code>
For items in list	<code>for x in alist:</code>
For indexes/values in a list	<code>for index,value in enumerate(alist):</code>
For each key in a dict	<code>for x in adict.keys():</code>
For all items in dict	<code>for key,value in adict.items():</code>
while <logic test>	do:
Exit loop immediately	<code>break</code>
Skip rest of loop and do loop again	<code>continue</code>

Logic and Math Operators

Math Operator	Example	X=7, Y=5
Addition	<code>X + Y</code>	12
Subtraction	<code>X - Y</code>	2
Multiplication	<code>X * Y</code>	35
Division	<code>X / Y</code>	1.4
Floor	<code>X // Y</code>	1
Exponent	<code>X ** Y</code>	16807
Modulo	<code>X % Y</code>	2

Logic Operator

Equality	<code>X == Y</code>	False
Greater Than	<code>X > Y</code>	False
Less Than	<code>X < Y</code>	True
Less or Equal	<code>X <= Y</code>	True
Not Equal	<code>X !=Y or X<>Y</code>	True

Logic and Math Operators (cont)

Bitwise Exclusive Or	<code>a ^ b</code> <code>xor(a, b)</code>
----------------------	--

Converting Data Types

Covert	Syntax	Example	Result
Num -> string	<code>str(number)</code> int, float or long	<code>str(100)</code> <code>str(3.14)</code>	'100' '3.14'
Encoded bytes -> string	<code>str(txt,encoding)</code>	<code>str(data,"utf8")</code>	string with data
Num String -> int	<code>int("string",base)</code> default base is 10	<code>int("42")</code> <code>int("101",2)</code> <code>int("ff", 16)</code>	42 5 255
int -> hex string	<code>hex(integer)</code>	<code>hex(255)</code> <code>hex(10)</code>	'0xff' '0xa'
integer -> binary string	<code>bin(integer)</code>	<code>bin(5)</code> <code>bin(3)</code>	'0b10' '0b11'
float -> integer	<code>int(float)</code> drops decimal	<code>int(3.14159)</code> <code>int(3.9)</code>	3 3
int or str -> float	<code>float(int or str)</code>	<code>float("3.4")</code> <code>float(3)</code>	3.4 3.0
String -> ASCII	<code>ord(str)</code>	<code>ord("A")</code> <code>ord("1")</code>	65 49
int -> ASCII	<code>chr(integer)</code>	<code>chr(65)</code> <code>chr(49)</code>	'A' '1'
bytes -> string	<code><bytes>.decode()</code>	<code>b'ABC'.decode()</code>	'ABC'
string -> bytes	<code><str>.encode()</code>	<code>'abc'.encode()</code>	b'abc'

Useful OS functions

import os

Executing a shell command	<code>os.system()</code>
Rename the file or directory src to dst	<code>os.rename(src, dst)</code>
Change working directory	<code>os.chdir(path)</code>
Get the users environment	<code>os.environ()</code>
Returns the current working directory	<code>os.getcwd()</code>

Dictionaries

Create an empty dict	dict={}
Initialize a non-empty dictionary	dict= { "key": "value", "key2": "value2"}
Assign a value	dict["key"]="value"
Determine if key exists	"key" in dict
Access value at key	dict["key"], dict.get("key")
Iterable View of all keys	dict.keys()
Iterable View of all values	dict.values()
Iterable View of (key,value) tuples	dict.items()

Slicing and Indexing

x[start:stop:step]	x=[4,8,9,3,0]	x="48930"
x[0]	4	'4'
x[2]	9	'9'
x[:3]	[4,8,9]	'489'
x[3:]	[3,0]	'30'
x[:-2]	[4,8,9]	'489'
x[:2]	[4,9,0]	'490'
x[:-1]	[0,3,9,8,4]	'03984'
len(x)	5	5
sorted(x)	[0,3,4,8,9]	['0','3','4','8','9']

Misc

Adding Comments to code:

#Comments begin the line with a pound sign

Adding Multi-line Comment to code

```
" " "
```

Multi-Line Comment

```
" " "
```

Get user input from keyboard

```
name = input( "What is your name? ")
```

Functions

```
def add(num1, num2):
```

```
    #code blocks must be indented
```

```
    #each space has meaning in python
```

```
    myr esult = num1 + num2
```

```
    return myresult
```

if then else statements

```
if <logic test 1>:
```

Misc (cont)

```
> #code block here will execute
#when logic test 1 is True
elif <logic test 2>:
    #code block executes if logic test 1 is
    #False and logic test 2 is True
else:
    #else has no test and executes when if
    #and all elif are False
python3 shebang
#!/usr/bin/env python3
```

Printing

Standard print	print('{}'.format(STRING))
Print string variable	print ("I am %s" % name)
Print int variable	print ("I am %d" % number)
Print with +	print ("I am " + name)

String Operations

Make lowercase	"Ab".lower()=="ab"
Make UPPERCASE	"Ab".upper()=="AB"
Make Title Format	"hi world".title()=="Hi World"
Replace a substring	"123".replace('2','z')= "1z3"
Count occurrences of substring	"1123".count("1")=2
Get offset of substring in string	"123".index("2")=1
Detect substring in string	"is" in "fish" == True