Introduction to Python

INTRODUCTION

Data Structures p30

- List [a,b,a,c] collection of items
- set {a,b,c} collection unique items
- tuple (a,b,c) immutable collection
- dictionary {key_1:val_1, key_2:val_2, …} pairs

LIST

A list is an *ordered* collection of objects

- x = [1, 7, 8, 3, 7]
- y = [7, 1, 3, 7, 8]

Objects can be extracted by their positional index

The index starts at position 0

- x[0] = 1
- x[1] = 7
- y[0] = 7
- y[2] = 3

INTRODUCTION – slicing a list

```
x = [1,3,5,8,2,4]
```

```
x
```

```
[1, 3, 5, 8, 2, 4]
```

show first 4

```
x[:4]
```

```
[1, 3, 5, 8]
```

show all beyond the first 4

```
x[4:]
```

```
[2, 4]
```

INTRODUCTION – slicing a list

```
x = [1,3,5,8,2,4]
```

```
x
```

```
[1, 3, 5, 8, 2, 4]
```

show first 4

```
x[:4]
```

```
[1, 3, 5, 8]
```

show all beyond the first 4

```
x[4:]
```

```
[2, 4]
```

```
x[1:3]
```

```
[3, 5]
```

show items with index 1 and 2

```
x[-1]
```

```
4
```

show last item

```
x[:-1]
```

```
[1, 3, 5, 8, 2]
```

show all but not the last one

INTRODUCTION – functions for lists

`append(x)` adds `x` to the end of the list

`count(x)` counts how many times `x` appears in the list

`extend(L)` adds the elements in list `L` to the end of the original list

`index(x)` returns the index of the first element of the list to match `x`

`insert(i, x)` inserts element `x` at location `i` in the list, moving everything else along

`pop(i)` removes the item at index `i`

`remove(x)` deletes the first element that matches `x`

`reverse()` reverses the order of the list

`sort()` we've already seen

All these functions work: in-place

INTRODUCTION – lists

```
x = [1,3,5,8,2,4]
```

```
x.append(9)
x
```

```
[1, 3, 5, 8, 2, 4, 9]
```

```
L = [0,5,8]
```

```
x.extend(L)
x
```

```
[1, 3, 5, 8, 2, 4, 9, 0, 5, 8]
```

```
x.count(5)
```

```
2
```

INTRODUCTION – lists

```
x = [1,3,5,8,2,4]
```

```
x.append(9)
x
```

```
[1, 3, 5, 8, 2, 4, 9]
```

```
L = [0,5,8]
```

```
x.extend(L)
x
```

```
[1, 3, 5, 8, 2, 4, 9, 0, 5, 8]
```

```
x.count(5)
```

```
2
```

```
# index of 8 (first time)
```

```
x.index(8)
```

```
3
```

```
# insert 6 in position 3
```

```
x.insert(3,6)
x
```

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

```
# deletes item in position 3
```

```
x.pop(3)
x
```

```
[1, 3, 5, 8, 2, 4, 9, 0, 5, 8]
```

INTRODUCTION – lists

```
x
```

```
[1, 3, 5, 8, 2, 4, 9, 0, 5, 8]
```

```
# remove 8 (first time only)
```

```
x.remove(8)
x
```

```
[1, 3, 5, 2, 4, 9, 0, 5, 8]
```

```
# reverse list x
```

```
x.reverse()
x
```

```
[8, 5, 0, 9, 4, 2, 5, 3, 1]
```

INTRODUCTION – lists

```
x
```

```
[1, 3, 5, 8, 2, 4, 9, 0, 5, 8]
```

```
# remove 8 (first time only)
```

```
x.remove(8)
x
```

```
[1, 3, 5, 2, 4, 9, 0, 5, 8]
```

```
# reverse list x
```

```
x.reverse()
x
```

```
[8, 5, 0, 9, 4, 2, 5, 3, 1]
```

```
# duplicate list x
```

```
y = x.copy()
y
```

```
[8, 5, 0, 9, 4, 2, 5, 3, 1]
```

```
x.sort()
x
```

```
[0, 1, 2, 3, 4, 5, 5, 8, 9]
```

INTRODUCTION

Python constructs p55-56

- iterator
- enumerate
- zip

INTRODUCTION – Creating lists

```
In [3]: # range is an iterator (no elements in it)

In [4]: range(10)
Out[4]: range(0, 10)

In [5]: range(0,10)
Out[5]: range(0, 10)

In [6]: # create a list using iterator

In [7]: list(range(0,10))
Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

INTRODUCTION – Creating lists

```
In [6]: # create a list using iterator

In [7]: list(range(0,10))
Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [3]: a = list(range(3,9))
        a
Out[3]: [3, 4, 5, 6, 7, 8]

In [5]: a[1]
Out[5]: 4

In [6]: # index starts at 0
```

INTRODUCTION – Creating lists

using for loop

```
L = []    # empty list
```

```
for n in range(12):
    L.append(n**2)
```

```
L
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

INTRODUCTION – Creating lists

using for loop

```
L = []   # empty list
```

```
for n in range(12):
    L.append(n**2)
```

```
L
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

using list comprehension

```
L=[i**2 for i in range(12)]
```

```
L
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

INTRODUCTION – Creating lists

using for loop

```
for n in range(12):
    L.append(n**2)
```

using list comprehension

```
L=[i**2 for i in range(12)]
```

INTRODUCTION – Creating lists

```
a = [i for i in range(20)]
a
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
# multiples of 3
```

```
a = [i for i in range(20) if i%3 == 0]
a
```

```
[0, 3, 6, 9, 12, 15, 18]
```

INTRODUCTION – Creating lists

```
a = [i for i in range(20)]
a
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
# multiples of 3
```

reminder

```
a = [i for i in range(20) if i%3 == 0]
a
```

```
[0, 3, 6, 9, 12, 15, 18]
```

INTRODUCTION – Arithmetic Operations

Operator		Description
+	add	Addition (e.g., 1 + 1 = 2)
-	subtract	Subtraction (e.g., 3 - 2 = 1)
*	multiply	Multiplication (e.g., 2 * 3 = 6)
/	divide	Division (e.g., 3 / 2 = 1.5)
//	floor_divide	Floor division (e.g., 3 // 2 = 1)
**	power	Exponentiation (e.g., 2 ** 3 = 8)
%	mod	Modulus/remainder (e.g., 9 % 4 = 1)

INTRODUCTION – Creating lists

```
list with 5 random integers
```

```
import random
```

```
random.seed(0)
```

```
x = [random.randint (0, 100) for i in range(5)]
x
```

```
[49, 97, 53, 5, 33]
```

FUNCTION

```
def f(x):
    return x**2
```

$y = x^2$

```
f(2.5)
```

```
6.25
```

LAMBDA FUNCTION

```
def f(x):
    return x**2
```

$y = x^2$

```
f(2.5)
```

```
6.25
```

```
# lambda function
```

```
g = lambda x:x**2
```

$y = x^2$

```
g(2.5)
```

```
6.25
```

INTRODUCTION – Apply a function to many values

```
def f(x):
    return x**2
```

```
f(2.5)
```

```
6.25
```

```
# lambda function
```

```
g = lambda x:x**2
```

```
g(2.5)
```

```
6.25
```

map

```
f = lambda x:2*x
```

y = 2x

```
f(2.5)
```

```
5.0
```

```
list(map(f,range(10)))
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

SET

A set is an *unordered* collection of unique objects

```
x = {1,2,4,5,2,5}
x
```

```
{1, 2, 4, 5}
```

(no duplicates)

```
type(x)
```

```
set
```

sets do not support indexing or slicing