UNIVERSITY of WISCONSIN
LA CROSSE
COMPUTER SCIENCE

CS 224 Introduction to Python

Map and Zip

1

An interesting problem…

Write a Python function called `xover` that takes the following parameters: one or two lists of bits, and a probability `indpb` (as a real number in [0.0, 1.0]). Let the first list be called `p1` and the second `p2`. `p2` and `indpb` are optional. Their default values are `None` and `0.5`, respectively. It does not return a value.

`xover` goes through `p1`, replacing each bit with the corresponding bit from `p2` with probability `indpb`, if `p2` was provided. Otherwise, bits in `p1` are replaced with 1 with probability `indpb`.

2

Solution 1

```
def xover(p1, p2=None, indpb=0.5):
    if p2 is None:
        p2 = [1] * len(p1)

    for i in range(len(p1)):
        if random() < indpb:
            p1[i] = p2[i]
```

3

Solution-ish 2 – using an iterator

```
def xover(p1, p2=None, indpb=0.5):
    if p2 is None:
        p2 = [1] * len(p1)

    for e in p1:
        if random() < indpb:
            e = <something>
```

IMPORTANT: changing e doesn't change the value in p1

Hmm, what goes here?

4

Solution-ish 3 – using an iterator

Problem: confusion about the difference between an index and a value:

```
def xover(p1, p2=None, indpb=0.5):
    if p2 is None:
        p2 = [1] * len(p1)

    for e in p1:
        if random() < indpb:
            p1[e] = p2[e]
```

e is a value in p1 NOT an index into p1!

5

Solution-ish 4 – comprehension anyone?

Could we use a comprehension? Wouldn't it be complicated?

Sort of and yes.

```
p1 = [p1[i] if random() < indpb else 1 if p2 is None
                    else p2[i] for i in range(len(p1))]
```

This works! Almost.

But it's a little complicated.

6

Unwinding Solution-ish 4

Let's break this down.

First: there is no filtering in this comprehension!

```
p1 = [p1[i] if random() < indpb else 1 if p2 is None
                                  else p2[i] for i in range(len(p1))]
```

NOT filtering.

7

An Aside

Quick review of the format of a comprehension:

```
p1 = [put f(e) in new list  for each e in old list  if filter]
```

filter goes here and **may affect number of elements** placed in the resulting list

8

Unwinding Solution-ish 4

```
p1 = [p1[i] if random() < indpb
      else 1 if p2 is None
      else p2[i]
```

What goes in new list

```
for i in range(len(p1))]
```

Iteration of old list

9

Unwinding Solution-ish 4

```
p1[i] if random() < indpb  else
1     if p2 is None        else
p2[i]
```

Mutually exclusive conditional logic to determine what is put in the new list

determines what is added
NOT if something is added

logically equivalent but not valid syntax

Condition	What to add
`if random() < indpb`	`add p1[i] to list`
`else if p2 is None`	`add 1 to list`
`else`	`add p2[i] to list`

10

A Simpler Example

Record coin flips

```
flips = [‘T’ if random() < 0.5 else ‘H’ for _ in range(20)]
```

11

Another Simpler Example

Another example: take even numbered elements from L1 and
odd numbered elements from L2, where
L1 and L2 have the same length

```
new = [L1[i] if i % 2 == 0 else L2[i]
                              for i in range(len(L1))]
```

12

Unwinding Solution-ish 4

Returning to our question:
Could we use a comprehension? Wouldn't it be complicated?

```
p1 = [p1[i] if random() < indpb else 1 if p2 is None
                    else p2[i] for i in range(len(p1))]
```

Why does this only "sort of" work?

The comprehension works. But to use it in our function, we have to reassign p1, thus we are no longer changing the list in the calling context.

13

Maps

In Python, a map provides another way to apply a function to each element of an iterable (list, tuple, etc.)

```
def convert(deg_c):
    return deg_c * 1.8 + 32

f_list = map(convert, c_list)
```

apply this function

to this list

14

Map Example 2

Let u_lists be a list of unsorted lists of integers. Use map to create a list containing the same sublists but with their elements in sorted order:

```
s_lists = map(sorted, u_lists)
```

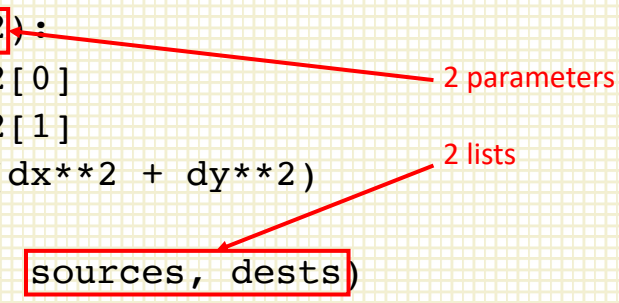
15

Map Example 3

Find distances between corresponding locations in a list of starting points and a list of destinations:

```
def distance(pt1, pt2):
    dx = pt1[0] – pt2[0]
    dy = pt1[1] – pt2[1]
    return math.sqrt(dx**2 + dy**2)

dists = map(distance, sources, dests)
```



16

Map Example 4

Using an ad hoc function applied to a list of integers:

```
polys = map(lambda x: 2*x + 4, int_list)
```

17

Zip

Create a list of tuples from some number of other lists:

```
result_tuples = zip(list1, list2, …, listn)
```

- result_tuples is a list
- each element is a tuple
- each tuple contains n elements – one from each list

18

Zip Example 1

```
digits = [1, 2, 3]
words = ['one', 'two', 'three']
romans = ['i', 'ii', 'iii']

combos = zip(digits, words, romans)

combos: [(1, 'one', 'i'), (2, 'two', 'ii'),
         (3, 'three', 'iii')]
```

19

Zip Example 2

Take an unsorted list and create a list of tuples that contain the values and their position in the original list:

```
nums = unsorted list of n integers

order = zip(nums, [i for i in range(len(nums))])
sorted_order = sorted(order, key=lambda x: x[0])
```

20