A tiny bit more Python

Ruth Anderson
UW CSE 160
Autumn 2022

See in python tutor

Enumerate a list

```
lst = [10 ** x for x in range(10)]

for i in range(len(lst)):
    print('val at index', i, 'is', lst[i])
```

$$\texttt{print('val at index', }\underbrace{\texttt{i,}}_{\text{index}}\texttt{ 'is', }\underbrace{\texttt{lst[i]}}_{\text{val}}\texttt{)}$$

Or:

```
for index, val in enumerate(lst):
     print('val at index', index, 'is', val)
```

Like dict.items()

Enumerate a list

Goal: add each element's index itself

```
lst = [x for x in range(10)]
new_lst = []
for i, v in enumerate(lst):
      new_lst.append(i + v)
```

With a list comprehension:

```
lst = [x for x in range(10)]
new_lst = [i + v for i, v in enumerate(lst)]
```

See in python tutor

Activity: Enumerate a list

Goal: Given a list of participants, in the order they finished a race, create a dictionary that maps their name to their finishing place.

```
racers = ['Dino', 'Wilma', 'Barney', 'Fred']
➔ race_dict = {'Dino':1, 'Wilma':2, 'Barney':3, 'Fred':4}
```

With a loop:

With a dictionary comprehension:

```
race_dict = {key: value for <item> in <sequence>}
```

See in python tutor

Activity: Enumerate a list

Goal: Given a list of participants, in the order they finished a race, create a dictionary that maps their name to their finishing place.

```
racers = ['Dino', 'Wilma', 'Barney', 'Fred’]
➔ race_dict = {'Dino':1, 'Wilma':2, 'Barney':3, 'Fred':4}
```

With a loop:

```
race_dict = {}
for index, val in enumerate(racers):
    race_dict[val] = index + 1
```

With a dictionary comprehension:

```
race_dict = {key: value for <item> in <sequence>}
```

```
race_dict = {val: index + 1 for index, val in enumerate(racers)}
```

Ternary Assignment Motivation

A common pattern in python

```
if x > threshold:
     flag = "Over"
else:
     flag = "Under"
```

Or

```
flag = "Under"
if x > threshold:
     flag = "Over"
```

See in python tutor

Ternary Assignment

A common pattern in python

```
if x > threshold:
    flag = "Over"
else:
     flag = "Under"
```

With a ternary expression:

```
flag = "Over" if x > threshold else "Under"
```

Ternary Expression
"Three elements”

Ternary Assignment

$$\texttt{flag = }\underbrace{\texttt{"Over"}}_{\text{Result if true}}\ \underbrace{\texttt{if x > threshold}}_{\text{Condition}}\ \texttt{else}\ \underbrace{\texttt{"Under"}}_{\text{Result if false}}$$

- Only works for single expressions as results.
- Only works for if and else (no elif)

```
flag = <result if True> <Condition> else <result if False>
```

See in python tutor

Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```
lst = []
for i in range(8):
    if i % 2 == 0:
        lst.append('even')
    else:
        lst.append('odd')
```

or

```
lst = []
for i in range(8):
    lst.append(                              )
```

```
flag = <result if True> <Condition> else <result if False>
```

See in python tutor

Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```
lst = []
for i in range(8):
    if i % 2 == 0:
        lst.append('even')
    else:
        lst.append('odd')
```

or

```
lst = []
for i in range(8):
    lst.append('even' if i % 2 == 0 else 'odd')
```

See in python tutor

Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```
lst = []
for i in range(8):
    if i % 2 == 0:
        lst.append('even')
    else:
        lst.append('odd')
```

or

```
lst = []
for i in range(8):
    lst.append('even' if i % 2 == 0 else 'odd')
```

Or with a list comprehension!

```
lst = [<expression> for <item> in <sequence>]
```

See in python tutor

Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```
lst = []
for i in range(8):
    if i % 2 == 0:
        lst.append('even')
    else:
        lst.append('odd')
```

or

```
lst = []
for i in range(8):
    lst.append('even' if i % 2 == 0 else 'odd')
```

Or with a list comprehension!

```
lst = [<expression> for <item> in <sequence>]
```

```
lst = ['even' if i % 2 == 0 else 'odd' for i in range(8)]
```

Get more practice

List Comprehensions:

```
[(x, y) for x in seq1 for y in seq2 if
                              sim(x, y) > threshold]
```

Enumerate:

```
for index, value in enumerate(seq):
    …
```

Ternary If Statement:

```
flag = "Over" if x > threshold else "Under"
```