A tiny bit more Python

Rob Thompson
UW CSE 160
Winter 2021

Enumerate a list

```
the_list = [10 ** i for i in range(10)]
for i in range(len(the_list)):
    print(str(i) + ': ' + str(the_list[i]))
```

index

value

Or:

```
for index, value in enumerate(the_list):
     print(str(index) + ': ' + str(value))
```

Like dict.items()

Enumerate a list

Goal: add each element's index itself

```
the_list = list(range(10))
new_list = []
for i, v in enumerate(the_list):
      new_list.append(i + v)
```

With a list comprehension:

```
the_list = list(range(10))
new_list = [i + v for i, v in enumerate(the_list)]
```

Ternary Assignment

A common pattern in python

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

Or

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

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

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

Result if true

Condition

Result if false

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

Ternary Assignment

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

```
the_list = []
for i in range(16):
    if i % 2 == 0:
        the_list.append('even')
    else:
        the_list.append('odd')
```

or

```
the_list = []
for i in range(16):
    the_list.append('even' if i % 2 == 0 else 'odd')
```

Ternary Assignment

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

```
the_list = []
for i in range(16):
    if i % 2 == 0:
        the_list.append('even')
    else:
        the_list.append('odd')
```

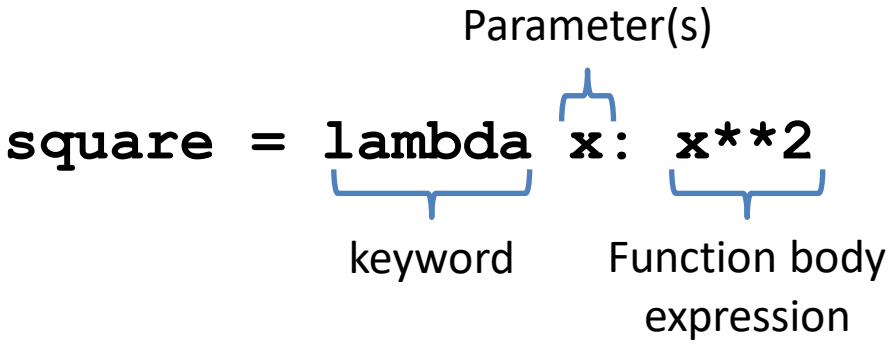
Or with a list comprehension!

```
the_list = ['even' if i % 2 == 0 else 'odd' for i in range(16)
```

Lambda Function

Functions take up space, and all have to be uniquely named

Solution: Make a function with no name!




```
square(9) # == 81
square(2) # == 4
```

Lambda Function

Without having to `def` them, we can create functions in a single line

```
nums = [5127, 6918, 9199, 57251]

def lastDigit(x):
    return x % 10
nums.sort(key=lastDigit)
```

vs

```
nums.sort(key=lambda x: x % 10)
```

Lambda Function

Can be used inside of loops and in list comprehensions, but the syntax is a little trickier

```
nums = [5127, 6918, 9199, 57251]

add_x = [(lambda x, n=n: x + n) for n in nums]


add_x[1](10) # == 6928
```

Get more practice

Enumerate:

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

Ternary If Statement:

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

Lambda Function:

```
add_two = (lambda x: x + 2)
```