

# CSc 120

## Introduction to Computer Programming II

### 14: List Comprehensions

# List comprehensions

- A *list comprehension* is a simple and concise way to create lists

# List comprehensions

- A *list comprehension* is a simple and concise way to create lists

Example: compute a list of squares of numbers

```
>>> def squares(n):  
    outlist = []  
    for i in range(n):  
        outlist.append(i*i)  
    return outlist  
  
>>> squares(3)  
[0, 1, 4]  
>>> squares(6)  
[0, 1, 4, 9, 16, 25]  
>>>
```

```
>>> def squares1(n):  
    return [i*i for i in range(n)]  
  
>>> squares1(3)  
[0, 1, 4]  
>>> squares1(6)  
[0, 1, 4, 9, 16, 25]  
>>>
```

list comprehension

# Basic structure

**[ *expr* **for** *item* **in** *some\_list* **if** *cond* ]**

|||

**new\_list = []**

**for *item* in *some\_list*:**

***if* *cond*:**

**new\_list.append(*expr*)**

**filter  
(optional)**

# Example 1

```
>>> def odds_and_evens(arglist):
    odds = []
    evens = []
    for i in range(len(arglist)):
        if i % 2 == 0:
            evens.append(arglist[i])
        else:
            odds.append(arglist[i])
    return (odds, evens)

>>> (o,e) = odds_and_evens([0,1,2,3,4,5])
>>> o
[1, 3, 5]
>>> e
[0, 2, 4]
>>>
```

# Example 1

```
>>> def odds_and_evens(arglist):
    odds = []
    evens = []
    for i in range(len(arglist)):
        if i % 2 == 0:
            evens.append(arglist[i])
        else:
            odds.append(arglist[i])
    return (odds, evens)
```

using list comprehensions

```
>>> def odds_and_evens(arglist):
    idxs = range(len(arglist))
    odds = [arglist[i] for i in idxs if i % 2 != 0]
    evens = [arglist[i] for i in idxs if i % 2 == 0]
    return (odds, evens)
```

```
>>> (o,e) = odds_and_evens([0,1,2,3,4,5])
>>> o
[1, 3, 5]
>>> e
[0, 2, 4]
>>>
```

# EXERCISE

Use list comprehensions to create the following lists:

- a) A list of even numbers from 2 to 100
  
- a) The list of elements of arglist that are divisible by 3

## Example 2

```
>>> import string
>>>
>>> #strip punctuation from around words in a list
>>> def strip_punctuation(wordlist):
    puncts = string.punctuation
    return [wd.strip(puncts) for wd in wordlist]
>>> wordlist = "Look!  !!Here's--> punctuation: !@#$%".split()
>>> wordlist
['Look!', '!!Here's-->', 'punctuation: !@#$%']
>>>
>>> strip_punctuation(wordlist)
['Look', 'Here's', 'punctuation']
```

# EXERCISE

Use a list comprehension to create a list of the elements of alist that are at index positions that are divisible by 3.

# EXERCISE-sol

Use a list comprehension to create a list of the elements of alist that are at index positions that are divisible by 3.

```
[ alist[i] for i in range(len(alist)) if i % 3 == 0]
```

style considerations

# Style considerations

- Use loops for:
  - code that has side effects, i.e., does I/O or modifies other objects
- Use list comprehensions for:
  - creating lists
    - using code that does not have side effects
- Avoid long or nested list comprehensions
  - these can be hard to read and understand

```
def primes_upto(n):  
    return [prime for prime in range(2, n) if prime not in \  
            [notAPrime for i in range(2, int(n**0.5)) for notAPrime in range(i * 2, n, i)]]
```



# Exercise – ICA-40

Do all problems.