

CSC148-Section:L0301/L0401

Week#5-Friday

Instructed by

AbdulAziz Al-Helali

a.alhelali@mail.utoronto.ca

Office hours: Wednesday 11-1, BA2230.

Slides adapted from Professor Danny Heap course material
winter17

idiomatic python

- Writing code the **python way**.
- Python is **more flexible** than the community you are coding in. Try to figure out what the python way is:
 - **don't re-invent** the wheel (except for academic exercises),
 - e.g. **sum, set**
 - **use comprehensions** when you mean to produce a new list
 - (tuple, dictionary, set, . . .)
 - use **ternary** if when you want an expression that evaluates in different ways, depending on a condition

idiomatic python

- Why?
 - Other Python programmers will be able to understand your code.
 - Also, you will understand theirs
 - Saves your time.
 - Easier to read and efficient

idiomatic python

- idiomatic python
 - <http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html#references>

idiomatic python

- Do you have patience to write a code do the following if python did it for you?
 - Slicing a string
 - “word”[1:3]
 - Find sum of a L=[1,2,3,4,5]
 - sum(L)
- We will see more examples next.

Ternary Operators or conditional expressions

```
is_p1_turn = True  
player_name = 'p1' if is_p1_turn else 'p2'  
print(player_name)
```

list comprehensions

add (cubes of) first 10 natural numbers

- Write a code that add (cubes of) first 10 natural numbers

list comprehensions

add (cubes of) first 10 natural numbers

- You'll be generating a new list, so use a comprehension
- You want to add all the numbers in the resulting list, so use sum

```
print (sum ( [n**3 for n in range (10) ] ) )
```


Nested list comprehensions

add (cubes of) first 10 natural numbers

- What is the output of the following code:

```
print ( [sum ( [n**3 for n in range (m) ] ) for m in range (10) ] )
```

Nested list comprehensions

add (cubes of) first 10 natural numbers

- What is the output of the following code:

```
print ( [sum ( [n**3 for n in range (m) ] ) for m in range (10) ] )
```

[0, 0, 1, 9, 36, 100, 225, 441, 784, 1296]

Euclidean distance

- Given $L=[1,4,12]$
- Write a function to find the Euclidean distance:
- $d = \sqrt{x^2 + y^2 + z^2}$

```
from typing import List
def ecul_dist(x: List) -> float:
    """
    returns Euclidean distance from origin a given 3d point
    >>> x=[3,4,12]
    >>> ecul_dist(x)
    13.0
    """
```

Euclidean distance

- Given $L=[1,4,12]$
- Write a function to find the Euclidean distance:
- $d = \sqrt{x^2 + y^2 + z^2}$

```
from typing import List
def ecul_dist(x: List) -> float:
    """
    returns Euclidean distance from origin a given 3d point
    >>> x=[3,4,12]
    >>> ecul_dist(x)
    13.0
    """
    return sum([c**2 for c in x])**0.5
```



Distance for vectors of x and y

- Write a function that returns a list of distances from origin given two vectors x and y.

```
def distance(x: List, y: List) -> List:
    """
        returns a list of distances from origin a given
        two vectors x and y.
        >>> x=[0,3,5]
        >>> y=[0,4,12]
        >>> distance(x,y)
        [0.0, 5.0, 13.0]
    """
```

Distance for vectors of x and y

- Write a function that returns a list of distances from origin a given two vectors x and y.

```
def distance(x: List, y: List) -> List:
    """
        returns a list of distances from origin a given
        two vectors x and y.
        >>> x=[0,3,5]
        >>> y=[0,4,12]
        >>> distance(x,y)
        [0.0, 5.0, 13.0]
    """
    return [ (a**2+b**2) **0.5 for a,b in zip(x,y) ]
```



zip

```
scores = [89, 90, 99]
names = ['Alice', 'Pop', 'Trudy']

r = zip(scores, names, scores)
print(res)
# Converting r iterator to a set
res_set = set(res)
print(res_set)
```

Average Length of words in a list

- Given

`L = ['hello', 'welcome', 'hi']`

- Print average Length of words in L

Average Length of words in a list

- Given

```
L = ['hello', 'welcome', 'hi']  
print(sum(len(s) for s in L) / len(L))
```

- Print average Length of words in L

Searching for a phrase in a list of words

- Given

`L = ['hello', 'welcome', 'hi']`

- Print True if 'lo' in **any** of the word in L

Searching for a phrase in a list of words

- Given

```
L = ['hello', 'welcome', 'hi']  
print(any('lo' in s for s in L))
```

- Print True if 'lo' in **any** of the word in L

Searching for a phrase in a list of words

- Given

`L = ['hello', 'welcome', 'well']`

- Print True if 'el' in **all** of the words in L

Searching for a phrase in a list of words

- Given

```
L = ['hello', 'welcome', 'well']  
print(all('l' in s for s in L))
```

- Print True if 'el' in **all** of the words in L

list and sets

- python **lists** allow duplicates, python **sets** don't
- python **sets** have a **set-difference** operator
- python built-in functions `list()` and `set()` convert types

list and sets,
example: find common items

```
list1 = ['a', 'c', 'c', 'e', 'a']  
list2 = ['a', 'b']
```

list and sets,
example: find common items

```
list1 = ['a', 'c', 'c', 'e', 'a']  
list2 = ['a', 'b']  
set1 = set(list1)  
set2 = set(list2)  
print(set1.intersection(set2))  
{ 'a' }
```


list and sets

example: compare if two lists **has the same items**

```
list1 = ['a', 'c', 'c', 'e', 'a']  
list2 = ['a', 'e', 'e', 'e', 'e', 'c']  
print(set(list1) == set(list2))  
True
```

list and sets

example: compare if two lists **has the same items**

```
list1 = ['a', 'c', 'c', 'e', 'a']  
list2 = ['a', 'e', 'e', 'e', 'e', 'c']  
print(set(list1) == set(list2))  
True
```

Thank you