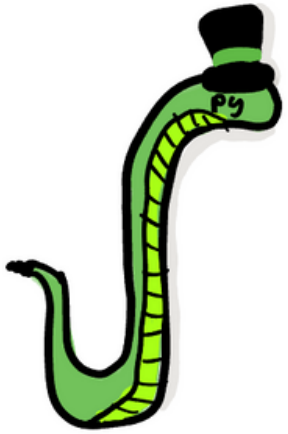


Misc. Topics Lecture

by Ecy!

Stanford | ENGINEERING
Computer Science
Ecy King, CS106A, 2023

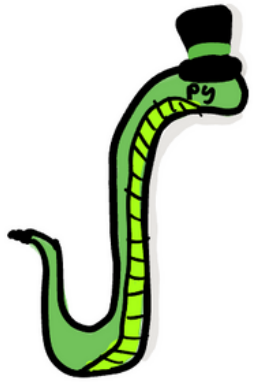
Housekeeping

- **Assignment 6, Bajillion is due Tues, Aug 15th at 11:59 pm**
  - with Grace Period until Wednesday, Aug 16th at 11:59 pm
- **No section next week**
- **Final a week from now, next week Friday, Aug 18th in here**
- **YOU'RE AT THE LAST ASSIGNMENT; LOOK AT HOW FAR YOU'VE COME :)!**

Photo Credits: Canva

Today

- **Tuples**
  - Exploring a new iterable type
- **Lambdas, Maps, and Sorting**
  - Exploring cool stuff with iterables
- **Some Shortcuts**
  - List Comprehensions
  - If-else statements and the Ternary Operator
- **Making Our Own Projects**
  - How can we use a template to explore programming?
  - How do we create our own start-to-finish project in PyCharm?



Tuples: a New Type

Overview of Tuple Functionality

- **Declaring a tuple**
  - *Tuples are declared as a bundle of parentheses.*
    - `my_tup = ('Fresno', 93720,  'CA')`

- **Accessing Values**
  - *We can access and store tuple values with zero-indexing.*
    - `city_value = my_tup[0]`

- **Updating Values**
  - *Tuples are **immutable!** We can't update nor modify the values directly.*
    - ~~`my_tup[0] = 'Clovis'`~~
    - ~~`my_tup[1] += 1`~~

- **Operations**
  - `len(my_tup)` `# evaluates to 3`
  - `my_tup[2]` `# can index into`

Tuples

- Groups a few items together
- Known number of elements
- Elements may be different types
- Examples
  - Ordered pairs:
    - (3, 4, 5)
  - Name & SunetID number
    - ('ecyfemi', 0314159)
  - RGB values
    - (255, 0, 255)

Lists

- Many elements; may append
- Unknown number of elements
- Typically same type
- Examples
  - Store urls
    - ['www.stanford.com', 'tinyurl.com']
  - Average temperatures
    - [96.7, 99, 98.2]

An Example

Tuples are declared as a bundle of parentheses.

Some characteristics are...

- *fixed size*
- *typically small*
- *sometimes different types inside*

- `show_rating = ('Miraculous Ladybug', 4.5)`
- `show = show_rating[0]`
- `rating = show_rating[1]`
- `len(show_rating)` `# would be 2`
- `len(show_rating[0])` `# would be 18`
- ~~`rating += 0.1`~~

Looping

We can also loop through the items in a tuple.

```
show_rating = ('Miraculous Ladybug', 4.5)
for item in show_rating:
    print(item)

"""
prints
    Miraculous Ladybug
    4.5
"""
```

Lists of Tuples

Often, we'll have to deal with lists of tuples.

```
show_ratings = [('Miraculous Ladybug', 4.5),
('Supa Team 4', 4.9), ('Naruto Shippuden',
4.7)]

# prints ('Naruto Shippuden', 4.7)
print(show_ratings[2])
# prints 4.7
print(show_ratings[2][1])
```

Lists of Tuples

We can loop through the list.

```
show_ratings = [('Miraculous Ladybug', 4.5),
('Supa Team 4', 4.9), ('Naruto Shippuden',
4.7)]

for show_rating in show_ratings:
     print(show_rating[0], show_rating[1])
```

Lists of Tuples

A little secret! The function d.items() returns a list of tuples!

```
city_pops = {'Fresno': 500,000, 'Palo Alto',
67, 000, 'LA': 3,849,000}

for city_pop in city_pops.items():
     print(city_pop[0], city_pop[1])

"""
prints
 Fresno 500000
 Palo Alto 67000
 LA 3849000
"""
```

```
city_pops.items() is
[('Fresno', 500,000),
 ('Palo Alto', 67,000),
 ('LA', 3,849,000)]
```

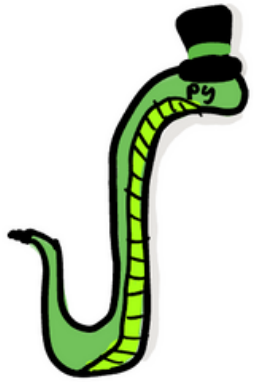
Returning Multiple Values

We can also use tuples to return multiple values :)

```
def get_origin()
    return 0,0
def calculate_distance():
    x,y = get_origin()
    print(x, y) # will print 0, 0
```

Overview of Tuple Functionality

- **Declaring a tuple**
  - `my_tup = ('Fresno', 93720,  'CA')`
- **Accessing Values**
  - `city_value = my_tup[0]`
  - `for item in my_tup:`
  - `    print(item)`
  - ~~`my_tup[0] = 'Clovis'`~~
- **Operations**
  - `len(my_tup)` `# evaluates to 3`
  - `my_tup[2]` `# can index into`
- **Dealing with Multiple Values**
  - `x, y = (4, 5)`
  - `return x, y`



The Lambdas, Maps, and Sorting!

The Iterables

Lists

```
my_list = []
my_list.append(3)

for item in my_list:
    print(item)
```

Strings

```
my_string = ""
my_string += "c"

for char in my_string:
    print(char)
```

Dictionaries

```
my_dict = {}
my_dict['key'] = 'value'

for key in my_dict:
    print(key, dict[key])
```

Tuples

```
my_tup = ()
my_tup = (3, 4, 5)

for elem in my_tup:
    print(elem)
```

The Map Function

Sometimes we want to do something with every element in an iterable. We can do this with the map function!

The map function takes in a function and an iterable.
This function gets applied to each ELEMENT of the iterable.

```
def double_num(num):
    return num*2
nums = [1, 2, 3]
new_list = map(double_num, nums)
```

The Map Function

Sometimes we want to do something with every element in an iterable. We can do this with the map function!

The map function takes in a function and an iterable.
This function gets applied to each ELEMENT of the iterable.

```
def double_num(num):
    return num*2
nums = [1, 2, 3]
new_list = map(double_num, nums)
```

The map returns a strange object though, so we actually need to cast it as a list!

The Map Function

Sometimes we want to do something with every element in an iterable. We can do this with the map function!

The map function takes in a function and an iterable.
This function gets applied to each ELEMENT of the iterable.

```
def double_num(num):
    return num*2
nums = [1, 2, 3]
new_list = list(map(double_num, nums))
```

The map returns a strange object though, so we actually need to cast it as a list!

The Map Function

Sometimes we want to do something with every element in an iterable. We can do this with the map function!

The map function takes in a function and an iterable.
This function gets applied to each ELEMENT of the iterable.

```
def double_num(num):
    return num*2
nums = [1, 2, 3]
new_list = list(map(double_num, nums))
# prints [2, 4, 6]
print(new_list)
```

Map vs Another Function

```
                def double_num(num):
                    return num*2

def double_list(nums):
    result = []
    for num in nums:
        result.append(double_num(num))
    return result


def double_list_w_map(nums):
    result = list(map(double_num, nums))
    return result
```

Lambda

Sometimes, instead of a whole function, we just want to pull a quickie!

A lambda is a single line of code that embodies the task of a function.

```
double_n = lambda n: n * 2
print(double_n(4)) # prints 8
```

For the map function, instead of putting in a function, we can put in a lambda for each element.

```
list(map(double_n, my_list)
list(map(lambda n: n*2, my_list))
```

Lambda Breakdown

Sometimes, instead of a whole function, we just want to pull a quickie!

A lambda is a single line of code that embodies the task of a function.

```
double_n = lambda n: n * 2
```

- n is the variable name
- the right defines what we are doing to the variable

What other cool things can we do with iterables?

Min/Max

Sometimes we want to get the minimum value of a list/tuple.

```
my_list = [3, 1, 4, 1]
my_tuple = (5, 9, 2, 6)
min(my_list) # is 1
min(my_tuple) # is 2
```

Othertimes, we want the max.

```
max(my_list) # is 4
max(my_tuple) # is 9
```

Sum

Bonus: gets the sum of all the elements in a list/tuple!

```
sum(my_list) # is 9
sum(my_tuple)# is 22
# gets the average!
sum(my_list)/len(my_list)
```

Sorted

Sorted generates a NEW list with the elements in ascending order.

```
result = sorted(my_list)
print(result) # prints [1, 1, 3, 4]
```

Min/Max

Sometimes we want to get the minimum value of a list/tuple.

```
my_list = [3, 1, 4, 1]
my_tuple = (5, 9, 2, 6)
min(my_list) # is 1
min(my_tuple) # is 2
```

Othertimes, we want the max.

```
max(my_list) # is 4
max(my_tuple) # is 9
```

Sum

Bonus: gets the sum of all the elements in a list/tuple!

```
sum(my_list) # is 9
sum(my_tuple)# is 22
# gets the average!
sum(my_list)/len(my_list)
```

Sorted

Sorted generates a NEW list with the elements in ascending order.

```
result = sorted(my_list)
print(result) # prints [1, 1, 3, 4]
```

We can REVERSE the list so that the elements are in descending order.

```
result2 = sorted(my_list, reverse=True)
print(result2) # is [ 4, 3, 1, 1]
```

What if we want to sort our own way?

Keys

We can define our own criteria for sorting. Function applied for each element.

key = criteria to compare elements; lower elements sorted first

Examples:

- **Sort list in abc order?**
- **Sort list of strings by length?**
- **Sort list of numbers by absolute value**

Keys

We can define our own criteria for sorting. Function applied for each element.

key = criteria to compare elements; lower elements sorted first

Examples:

- **Sort list in abc order?**
  - `words = ['bacon', 'apple', 'cabbage']`
  - `result = sorted(words) # yay!`
- **Sort list of strings by length?**
- **Sort list of numbers by absolute value**

Keys

We can define our own criteria for sorting. Function applied for each element.

key = criteria to compare elements; lower elements sorted first

Examples:

- **Sort list in abc order?**
  - `words = ['banana', 'apple', 'clementine']`
  - `result = sorted(words) # yay!`
- **Sort list of strings by length?**
  - **`result = sorted(words, key=lambda w: len(w))`**
  - **`# result is ['apple', 'banana', 'clementine']`**
- **Sort list of numbers by absolute value?**

Keys

We can define our own criteria for sorting. Function applied for each element.

key = criteria to compare elements; lower elements sorted first

Examples:

- **Sort list in abc order?**
  - `words = ['banana', 'apple', 'clementine']`
  - `result = sorted(words) # yay!`
- **Sort list of strings by length?**
  - `result = sorted(words, key=lambda w: len(w))`
  - `# result is ['apple', 'banana', 'clementine']`
- **Sort list of numbers by absolute value?**
  - **`nums = [3, 1, 4, -1, -5,-9]`**
  - **`result = sorted(nums, key= lambda num: abs(num))`**
  - **`# result is [1, -1, 3, 4, -5, -9]`**

More Examples of Keys + Reverse

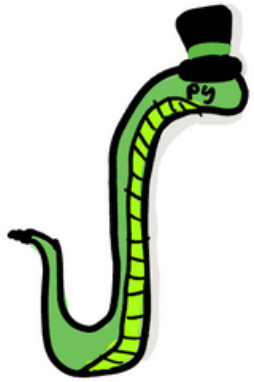
We can define our own criteria for sorting. Function applied for each element.

key = criteria to compare elements; lower elements sorted first

Examples:

- **Sort list in reverse abc order?**
  - `words = ['banana', 'apple', 'clementine']`
  - `result = sorted(words, reverse=True) # yay!`
- **Sort list of strings by longest length?**
  - `words = ['banana', 'apple', 'clementine']`
  - `result = sorted(words, key=lambda w: len(w), reverse=True)`

```
# result is ['clementine', 'banana', 'apple']
```



Some Shortcuts

S'more Shortcuts

We know how to do things the long way, now for the shortcut!

List Comprehensions

- creating a new list based on an old one

Condensing Ifs

Ternary Operator

- how can we do certain if-else statements on one line?

What if we want to make a list/perform an action based on an original list?

List Comprehensions

What if we want to make a list based on other list?

```
old_list = [3, 1, 4, 1, 5, 9]
new_list = [elem*2 for elem in old_list]
```

List Comprehensions

What if we want to make a list based on other list?

```
old_list = [3, 1, 4, 1, 5, 9]
new_list = [elem*2 for elem in old_list]
# same as
new_list = []
for elem in old_list:
    new_list.append(elem*2)
```

List Comprehensions

What if we want to make a list based on other list?

```
old_list = [3, 1, 4, 1, 5, 9]
new_list = [elem*2 for elem in old_list]
# same as
new_list = []
for elem in old_list:
    new_list.append(elem*2)

# new_list is [6, 2, 8, 2, 10, 18]
```

List Comprehensions

+filtering!

```
old_list = [3, 1, 4, 1, 5, 9]
new_list =
    [elem*2 for elem in old_list if elem % 2 == 0 ]
```

List Comprehensions

+filtering!

```
old_list = [3, 1, 4, 1, 5, 9]
new_list =
    [elem*2 for elem in old_list if elem % 2 == 0 ]

#same as
new_list = []
for elem in old_list:
   if elem % 2 == 0:
       new_list.append(elem*2)
```

List Comprehensions

+filtering!

```
old_list = [3, 1, 4, 1, 5, 9]
new_list =
    [elem*2 for elem in old_list if elem % 2 == 0 ]

#same as
new_list = []
for elem in old_list:
   if elem % 2 == 0:
       new_list.append(elem*2)

# new_list is [8]
# doubles (and keeps) element only if condition
```

List Comprehensions

How do we do quick actions?

```
old_list = [3, 1, 4, 1, 5, 9]
[print(elem) for elem in old_list]


#same as
for elem in old_list:
    print(elem)
```

What if want to do quick if else?

Quick If-Return

moves around if-else to be a one-liner

Before

```
def is_legal(age):
    if age >= 18:
        return True
    else:
        return False
```

After

```
def is_legal(age):
    return age >= 18
```

Ternary Op

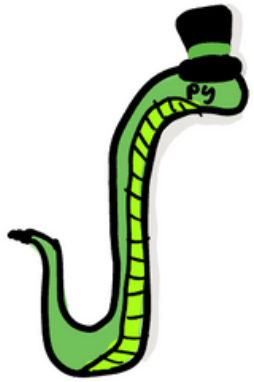
How can we spice up our python?

[action on true] if [expression] else [action on false]

```
# can do actions!
print('18+') if is_legal(x) else print('not yet')

# can store variables!
pixel.red =
    out_pixel.red if pixel.blue > 0.9*avg
        else pixel.red
```

*note that this can make code a lot more dense and should be used sparingly!



Writing Our Own Programs

How do we experiment with the code in our own ways?

Modify

Can take an assignment and mod it, basically doing an extension.

1) Copy a previous assignment
2) Add/Change stuff that suit your needs
3) Experiment & Enjoy!

Cool modifications in a limited space

Template

Can take an assignment, delete code, and do whatchya want

1) Copy a previous assignment
2) Delete stuff
3) Create the program you want

More experimentation and modification

Scratch

Do whatever we want, ground up

1) Open blank project or start from scratch
2) Create main boilerplate
3) Install necessary packages
3) Explore

Go wild at your own risk ;) LOTS of learning here!

Ideas to Explore

- Create an image filter that makes a black and white image
- Create a cool visualization (I made a DNA strand thingy)
- Create a game
- Create programs that process data relevant to you
- Explore interactive inputs with the input() and print() functions

Ideas to Explore

- Create an image filter that makes a black and white image
- Create a cool visualization (I made a DNA strand thingy)
- Create a game
- Create programs that process data relevant to you
- Explore interactive inputs with the input() and print() functions

```
user_input = input(prompt)
# takes in a prompt from the user as a string
(note: numbers will have to be cast as such)
```

Ideas to Explore

- Create an image filter that makes a black and white image
- Create a cool visualization (I made a DNA strand thingy)
- Create a game
- Create programs that process data relevant to you
- Explore interactive inputs with the input() and print() functions

Images Strings Data Graphics Classes

Lists Dictionaries Animation

Programming learned by exploring!

PyCharm is your Oyster!

PyCharm is your Oyster!

Blank Project on CS106A page!

Recap

Today, we talked about and learned how to...

Tuples and Iteratables

- how to create and manage tuples
- dealing with the iterables
- how to map and sort iterables
- how to use lambdas
- using list comprehensions
- using the ternary operator and shortening if-else statements

Independent Projects

- modifying existing assignments (aka creating your own extensions)
- using assignments as templates
- starting from scratch
  - absolute scratch
  - opening a blank project
- input function


THE END