On your way in…

Pick-up:

1. HW04, due Monday
2. POGIL Activity 30: Lambda

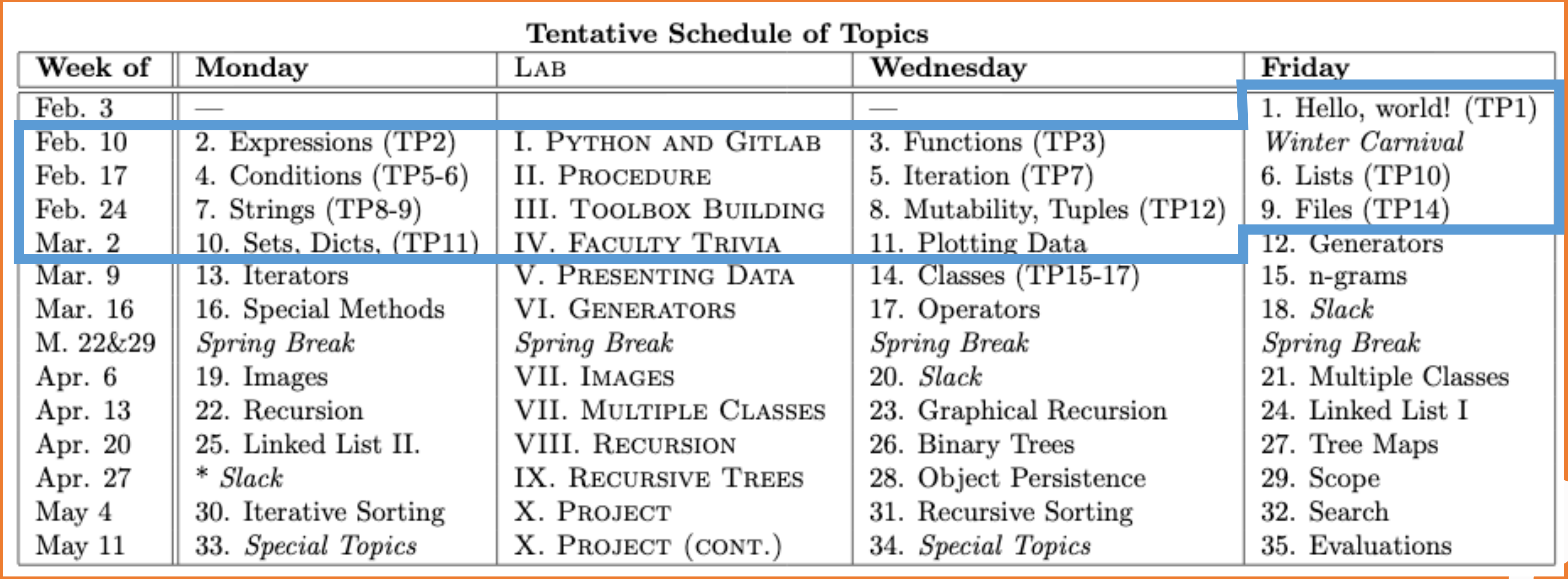
Midterm Exam is Thursday, March 12

- TPL 203
- 5:45pm-7:45pm OR 8-10pm
- Exam Review Session: 3/9 at 6-8pm in TPL 203.
- Closed book exam
- Review your homeworks! POGILs! Slides! Labs!

- Next week’s lab will be less intense

Midterm Exam is Thursday, March 12

Topic Coverage



Tentative Schedule of Topics

Week of	Monday	Lab	Wednesday	Friday
Feb. 3	—		—	1. Hello, world! (TP1)
Feb. 10	2. Expressions (TP2)	I. Python and Gitlab	3. Functions (TP3)	*Winter Carnival*
Feb. 17	4. Conditions (TP5-6)	II. Procedure	5. Iteration (TP7)	6. Lists (TP10)
Feb. 24	7. Strings (TP8-9)	III. Toolbox Building	8. Mutability, Tuples (TP12)	9. Files (TP14)
Mar. 2	10. Sets, Dicts, (TP11)	IV. Faculty Trivia	11. Plotting Data	12. Generators
Mar. 9	13. Iterators	V. Presenting Data	14. Classes (TP15-17)	15. n-grams
Mar. 16	16. Special Methods	VI. Generators	17. Operators	18. *Slack*
M. 22&29	*Spring Break*	*Spring Break*	*Spring Break*	*Spring Break*
Apr. 6	19. Images	VII. Images	20. *Slack*	21. Multiple Classes
Apr. 13	22. Recursion	VII. Multiple Classes	23. Graphical Recursion	24. Linked List I
Apr. 20	25. Linked List II.	VIII. Recursion	26. Binary Trees	27. Tree Maps
Apr. 27	* *Slack*	IX. Recursive Trees	28. Object Persistence	29. Scope
May 4	30. Iterative Sorting	X. Project	31. Recursive Sorting	32. Search
May 11	33. *Special Topics*	X. Project (cont.)	34. *Special Topics*	35. Evaluations

Midterm Exam is Thursday, March 12

Topic Coverage

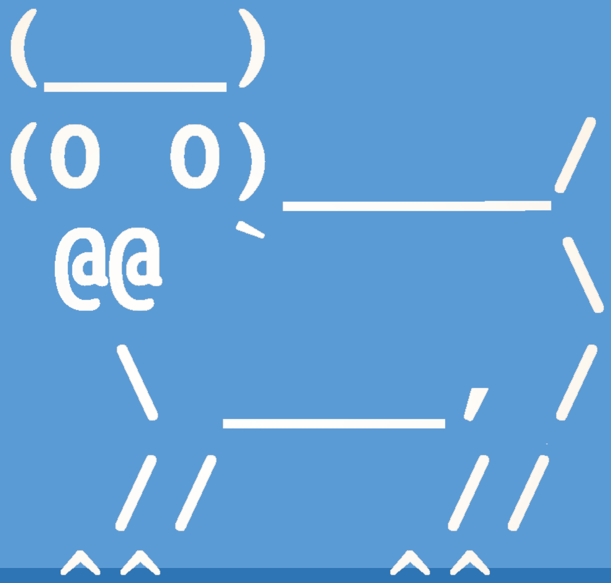
- *Homework 1*: Expressions & Functions, return & print
- *Homework 2*: booleans & loops over sequences, simplifying conditionals, list indexing
- *Homework 3*: strings & mutability
- *Homework 4*: Tuples, Dict (get), list comprehension, lambda sorting
- *From labs:*
    - Writing functions, File reading; Strip, split; Sorting, strings; Len; Finding max; Counters in loops; Doctests, __all__, modules/scripts, if __name__==‘__main__’
- Pretty much everything up to and including Lab 4 & Homework 4

Welcome to CS 134!

Introduction to Computer Science

Iris Howley

-Lambda-



Spring 2020

Useful Tuples

How to swap?

```
>>> first = 'harry'
>>> second = 'potter'
>>> tmp = first
>>> first = second
>>> second=tmp
>>> first
'potter'
>>> second
'harry'
```

- With Tuples:

```
>>> first = 'harry'
>>> second = 'potter'
>>> first, second = second, first
>>> first
'potter'
>>> second
'harry'
```

Useful Tuples

```
>>> name = ['harry','james','potter']
```

Storing list values?

```
>>> first = name[0]
>>> second = name[1]
>>> third = name[2]
>>> first
'harry'
>>> second
'james'
>>> third
'potter'
```

- With Tuples:

```
>>> first,second,third = name
>>> first
'harry'
>>> second
'james'
>>> third
'potter'
```

Sorting We've Seen Before

- `object.sort()`
  - Sorts object in-place (destroys original ordering)
  - Only makes sense for mutable objects, like a list
    - `myString.sort()` does **NOT** make sense, because strings are immutable

- `sorted(object)`
  - Returns a copy of object, sorted
  - We need to tie it to a balloon!
    - `sList = sorted(object)`

Sorting We’ve Seen Before

- `object.reverse()`
  - Reverse-sorts object in-place (destroys original ordering)
  - Only makes sense for mutable objects, like a list
    - `myString.reverse()` does **NOT** make sense, because strings are immutable

Any guesses about the default value of `reverse`?

- `sorted(object, reverse=True)`
  - Returns a copy of object, reverse-sorted
  - We need to tie it to a balloon!
    - `rsList = sorted(object, reverse=True)`

What happens when you call sorted without defining reverse?

`reverse=False`

TODAY'S LESSON

Sorting with Lambda

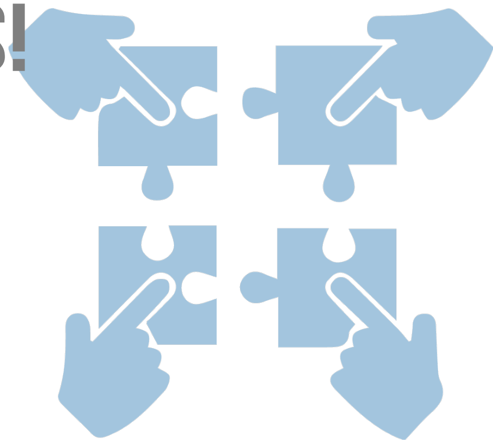
(Convenient ways to sort objects in customized ways)

POGIL Activity 23 - Lambda

- Look at Python Activity 23, Question 1-5
- Find a partner and talk through the questions together

PLEASE NOTE, THIS POGIL IS EXTRA FRESH OFF THE PRESSES AND MY HAVE SOME ERRORS!

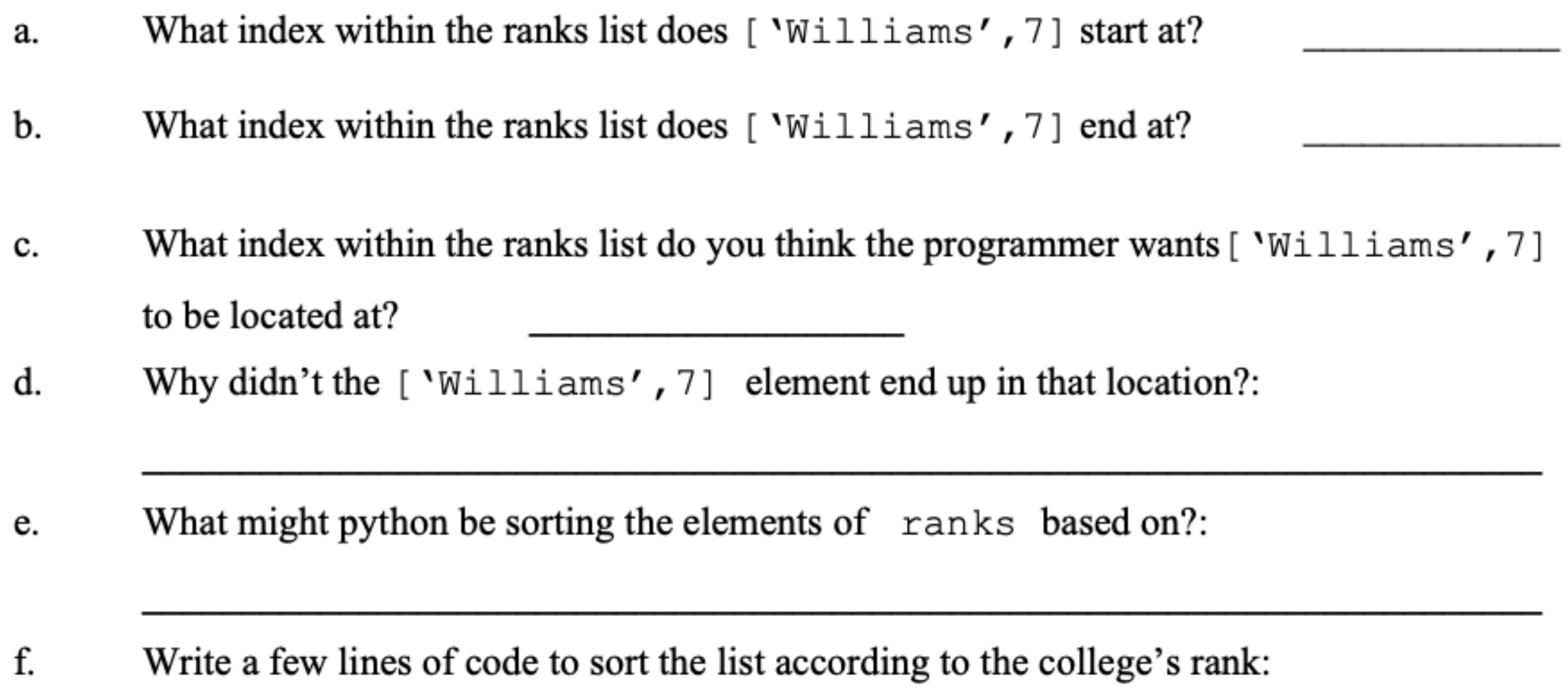
(Let me know if something seems off)



POGIL – Activity 23: Question 1

```
>>> ranks = [['Smith',18],['Williams',7],['Amherst',9]]
>>> sorted(ranks)
[['Amherst',9],['Smith,18],['Williams',7]]
```

THIS IS NOT THE DESIRED ORDERING!



a. What index within the ranks list does `['Williams',7]` start at? ____________

b. What index within the ranks list does `['Williams',7]` end at? ____________

c. What index within the ranks list do you think the programmer wants `['Williams',7]` to be located at? ________________

d. Why didn't the `['Williams',7]` element end up in that location?:

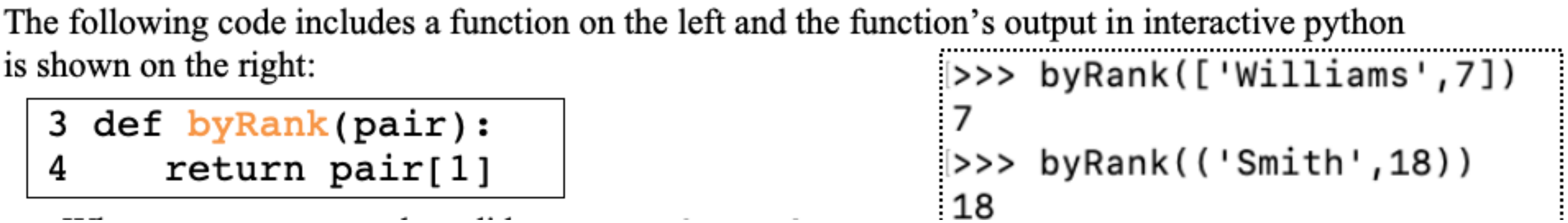
____

e. What might python be sorting the elements of `ranks` based on?:

____

f. Write a few lines of code to sort the list according to the college's rank:

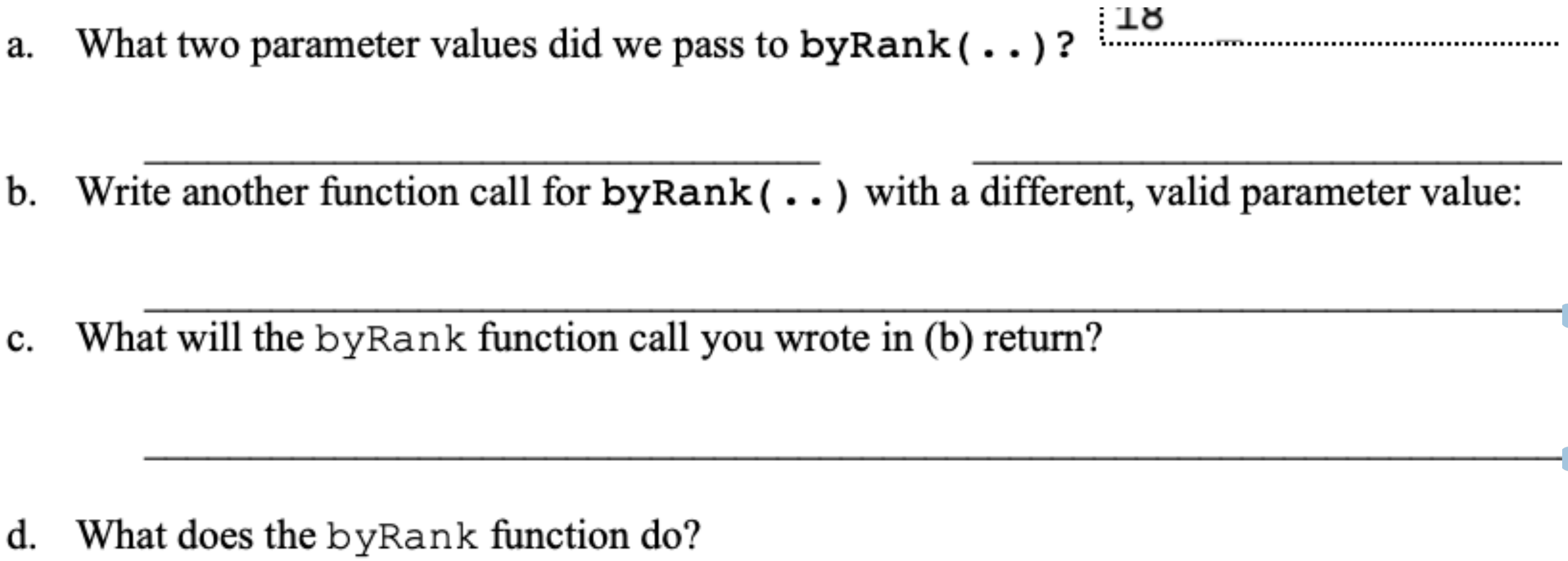
POGIL – Activity 23: Question 2



The following code includes a function on the left and the function's output in interactive python is shown on the right:

```python
def byRank(pair):
    return pair[1]
```

```
>>> byRank(['Williams',7])
7
>>> byRank(('Smith',18))
18
```



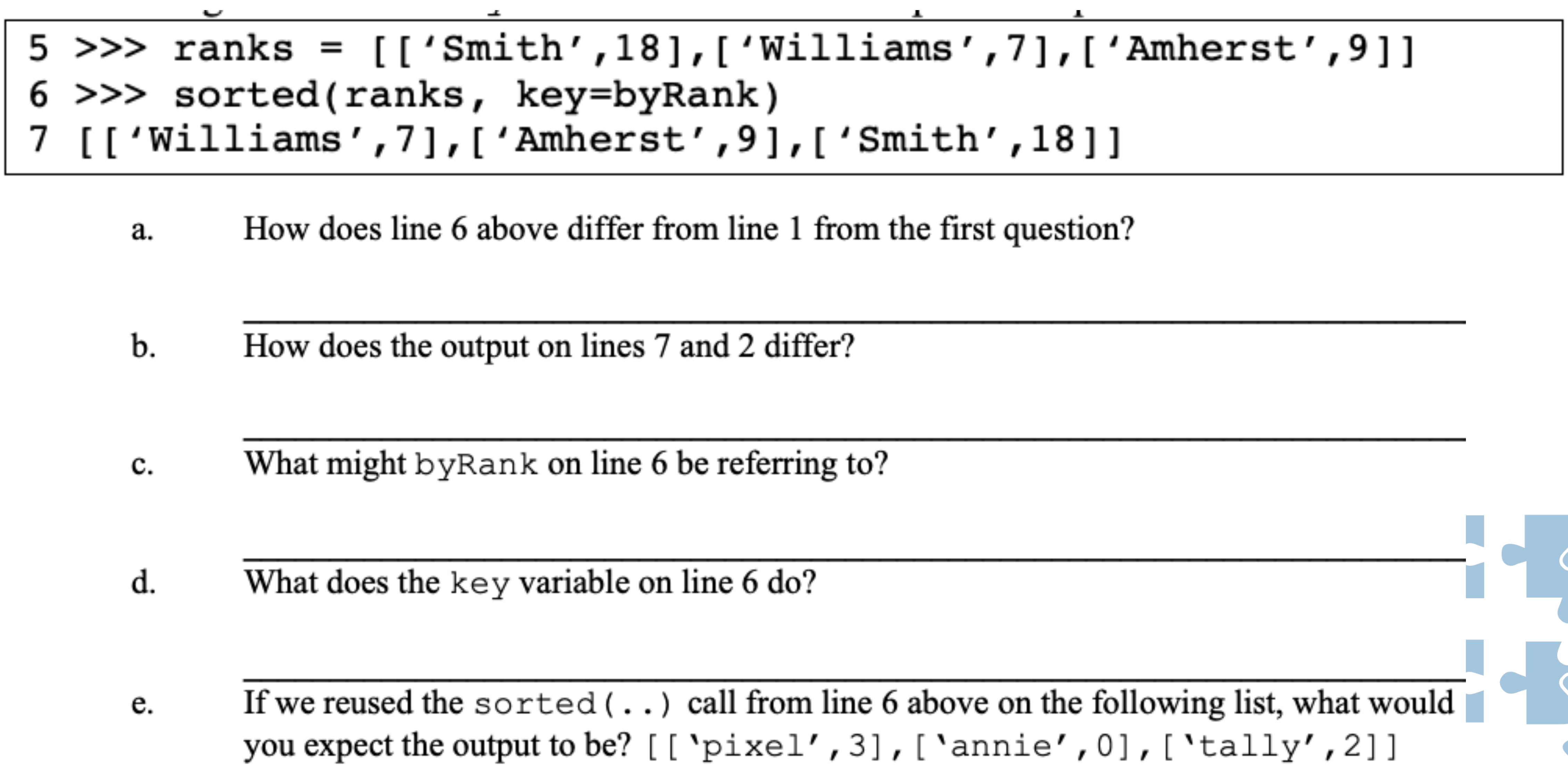
a. What two parameter values did we pass to **byRank(..)**?

b. Write another function call for **byRank(..)** with a different, valid parameter value:

c. What will the byRank function call you wrote in (b) return?

d. What does the byRank function do?

POGIL – Activity 23: Question 3



```
5 >>> ranks = [['Smith',18],['Williams',7],['Amherst',9]]
6 >>> sorted(ranks, key=byRank)
7 [['Williams',7],['Amherst',9],['Smith',18]]
```

a. How does line 6 above differ from line 1 from the first question?

b. How does the output on lines 7 and 2 differ?

c. What might `byRank` on line 6 be referring to?

d. What does the `key` variable on line 6 do?

e. If we reused the `sorted(..)` call from line 6 above on the following list, what would you expect the output to be? `[['pixel',3],['annie',0],['tally',2]]`

POGIL – Activity 23: Question 4

```
8 >>> ranks = [[‘Smith’,18],[‘Williams’,7],[‘Amherst’,9]]
9 >>> sorted(ranks, key=lambda pair:pair[1])
10 [[‘Williams’,7],[‘Amherst’,9],[‘Smith’,18]]
```

a. Examine the text that follows the `lambda` keyword on line 9 above, and the text of the `byRank` function in question 2. How do these differ?

____

b. How does the output on lines 10 and 7 differ?

____

c. What might the `key=lambda pair:pair[1]`on line 9 be doing?

____

d. If we changed line 9 to be `sorted(ranks, key=lambda pair:pair[0])` what might the output be?

____

e. The code in lines 8-9 above accomplishes the same tasks as the code in lines 3-6. Why might we use one approach over another?

Lambda Syntax

Denotes an unnamed function, we can't call it explicitly!

- `lambda x: x` A transformation, typically using the parameter we passed

  The variable that refers to the value we're being passed (like a parameter)

- `sqList=sorted(theList, key=lambda x: x*x)`

  `key` is an optional named parameter of `sorted(..)`

- What does `sqList` contain?

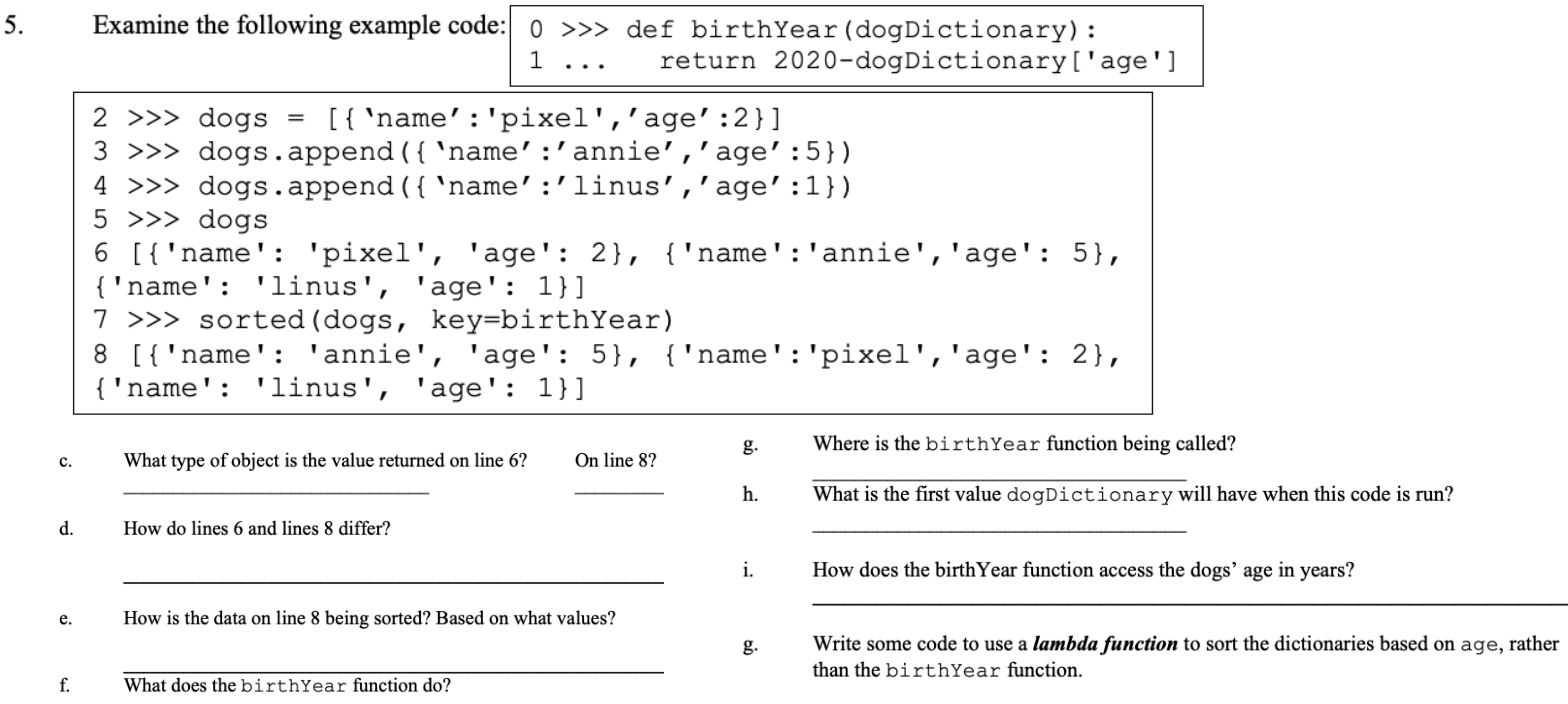
Lambda Syntax

- What happens when two values from lambda function are equal?
- Can also specify secondary sorting mechanism!

- `sortedCharacters = sorted (theList, key=lambda x: (x[1], x[2]))`

- Specifies what's in x[1] as primary sort key, and if there's equals, look at what's in x[2]

POGIL – Activity 23: Question 5



5. **Examine the following example code:**

```
0 >>> def birthYear(dogDictionary):
1 ...   return 2020-dogDictionary['age']
```

```
2 >>> dogs = [{‘name’:'pixel',’age’:2}]
3 >>> dogs.append({‘name’:’annie’,’age’:5})
4 >>> dogs.append({‘name’:’linus’,’age’:1})
5 >>> dogs
6 [{'name': 'pixel', 'age': 2}, {'name':'annie','age': 5},
{'name': 'linus', 'age': 1}]
7 >>> sorted(dogs, key=birthYear)
8 [{'name': 'annie', 'age': 5}, {'name':'pixel','age': 2},
{'name': 'linus', 'age': 1}]
```

c. What type of object is the value returned on line 6? ________________ On line 8? ________

d. How do lines 6 and lines 8 differ?

____

e. How is the data on line 8 being sorted? Based on what values?

____

f. What does the `birthYear` function do?

g. Where is the `birthYear` function being called?

____

h. What is the first value `dogDictionary` will have when this code is run?

____

i. How does the birthYear function access the dogs’ age in years?

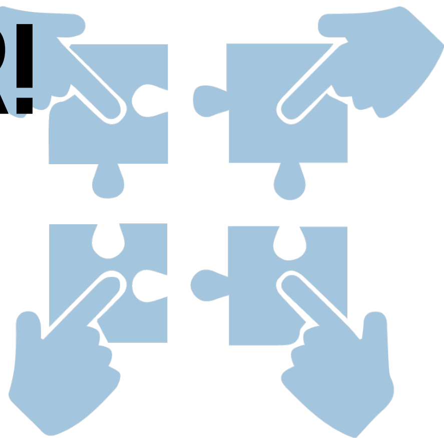
____

g. Write some code to use a ***lambda function*** to sort the dictionaries based on `age`, rather than the `birthYear` function.

YOU SHOULD COMPLETE THE REST OF ALL POGILS OUTSIDE OF CLASS.

BEST DONE WITH A PARTNER OR STUDY GROUP.

CHECK YOUR ANSWERS ON A COMPUTER!



TODAY'S LESSON

`Sorting with Lambda`

(Convenient ways to sort objects in customized ways)

An Example

- `>>> ranks = [('Amherst', 18), ('Williams', 7), ('Middlebury', 9)]`
- `>>> ranks.sort()`
- `>>> ranks`
- `[('Amherst', 18), ('Middlebury', 9), ('Williams', 7)]`

This isn’t what we want!

Customized Sorting

- What should we do?
  - Iterate through, find highest, insert at front of new list
  - Or maybe…use sorted() and its `key` parameter!

```
sorted(iterable[, key][, reverse])
```

Customized Sorting

- **`sorted(iterable[, key][, reverse])`**
- key should be a function that can be used for sort comparison
  - `ranks = [('Amherst', 18), ('Williams', 7), ('Middlebury', 9)]`

- `def byRank(pair):`
  - `return pair[1]`

- `rs = sorted(ranks, key=byRank)`

Sorting Tools

- `def byRank` is a simple**, one-expression** function with just this one purpose!
- …lambda functions (i.e. anonymous functions)

- `rl = sorted(ranks, key=lambda pair:pair[1])`
- Compare to:
- `rs = sorted(ranks, key=byRank)`
- `def byRank(pair):`
  - `return pair[1]`

Lambda Functions (Another Example)

- def mult(a,b):
  - return a*b

- p = mult(5,6)

- *Is comparable to:*
- m = lambda a,b: a*b

- p = m(5,6)

A poor use of lambda functions!

Lambda Functions (Another Example)

- Maybe we want to always transform a function's output in a couple different ways:
- `def somefunc(n):`
  - `return lambda a : a*n`
- `doubled = somefunc(2)`
- `print(doubled(5))` → `10`

Use lambda functions when an anonymous function is required for a short period of time

- `tripled= somefunc(3)`
- `print(tripled(5))` → `15`

Lambda Functions

- Historical significance to the field of computer science
- Introduced by Alonzo Church in the 1930s

- Thought they were writing about mathematical logic, ended up defining computation
  - ~1960s, connected lambda to programming languages
  - Popular in linguistics, too
    - See ‘Montague Grammar’
- Ties into Turing machines (~1935)
  - Defines an abstract machine
  - Proves fundamental limitations on the power of mechanical computation

EVERYTHING IN PYTHON IS AN OBJECT

(including functions)

QUESTIONS?

Leftover Slides

Functions as Objects

- `dogs = ['pixel', 'tally', 'linus', 'wally']`

What's happening here?

What if I wanted to use a different function

- `def justDog(d):`
  - `return d + " dog"`
- `def printDog(dList, strFunction):`
  - `for d in dList:`
    - `print(strFunction(d))`
- `>>> printDog(dogs, justDog)`

```
pixel dog
tally dog
linus dog
wally dog
```