

Name: _____

Partner: _____

Python Activity 39: Sorting with a Key Function**Learning Objectives**

Students will be able to:

Content:

- Explain how the **key=** named parameter impacts Python's **sorting** behavior
- Anticipate when it's appropriate to override Python's default **sorting** behavior

Process:

- Write code that sorts sequences by something other than the first element
- Write code that provides tie-breaking sorting behavior
- Write code that sorts sequences of *mixed types*

Prior Knowledge

- Python concepts: sequences, sorted(), ord(), type()

Critical Thinking Questions:

1. Examine the sample code below.

And sample data:

Python Program	Sample Data
<pre>def capacity(course_pair): '''Takes a sequence and returns item at index 1''' return course_pair[1]</pre>	<pre>courses = [['CS134', 74, 'Fa'], ['CS136', 60, 'Fa'], ['AFR206', 30, 'Spr'], ['ECON233', 30, 'Fa'], ['MUS112', 10, 'Fa'], ['STAT200', 50, 'Spr'],</pre>
<pre>sorted(courses, key=capacity)</pre>	

- a. What *type* of value is `courses`? **A list of** _____
- b. What might the *0th* inner element of `courses` represent? (i.e., 'CS134')

What might the *1th* inner element of `courses` represent? (i.e., 74)

What might the *2th* inner element of `courses` represent? (i.e., 'Spr')



- c. If we entered the following code, what might be the 0th element returned?
`sorted(courses)` The 1th element returned?



- d. When we run `sorted(courses, key=capacity)` we get the following output:

```
[['MUS112', 10, 'Fa'], ['AFR206', 30, 'Spr'], ['ECON233', 30, 'Fa'],
 ['STAT200', 50, 'Spr'], ['PSYC201', 50, 'Fa'], ['CS136', 60, 'Spr'],
 ['CS134', 74, 'Spr'], ['MATH110', 74, 'Spr']]
```

What's different about this function call compared to `sorted(courses)`?

How does Python determine the ordering of `courses` in this case?

What might the `key=capacity` named parameter do? (*Hint: What else is named "capacity" in the code above?*)

What *type* of object must follow the key named parameter? _____
e. If we wanted to sort courses based on the term a course is offered, how might we change this code?

2. Examine the following Python function, that continues from the previous example:

```
def term(course_pair):  
    '''Takes a sequence and returns item at index 2'''  
    return course_pair[2]
```

- a. What is different about the function `term`, as compared to the function `capacity`?
- b. What might the 0th element of a call to `sorted(courses, key=term)` be?
- _____

FYI: Python's sorting functions are *stable*, which means that items that are equal according to the sorting *key* have the same relative order as in the original sequence.

- c. Below is the output from `sorted(courses, key=term)`:

```
[['CS134', 74, 'Fa'], ['CS136', 60, 'Fa'], ['ECON233', 30, 'Fa'],  
 ['MUS112', 10, 'Fa'], ['PSYC201', 50, 'Fa'], ['AFR206', 30, 'Spr'],  
 ['STAT200', 50, 'Spr'], ['MATH110', 74, 'Spr']]
```



Why is the 'CS134' data the 0th item returned?

3. Examine the following Python function, that continues from the previous example:

```
def term_then_cap(course_pair):  
    '''???'''  
    return course_pair[2], course_pair[1]
```

- a. What is different about the function `term_then_cap`, as compared to the previous functions `capacity` and `term`?
- b. What might be the 0th element `sorted(courses, key=term_then_cap)` returns?
- c. Below is the output from `sorted(courses, key=term)`:

```
[['MUS112', 10, 'Fa'], ['ECON233', 30, 'Fa'], ['PSYC201', 50, 'Fa'],  
 ['CS136', 60, 'Fa'], ['CS134', 74, 'Fa'], ['AFR206', 30, 'Spr'],  
 ['STAT200', 50, 'Spr'], ['MATH110', 74, 'Spr']]
```



Why is the 'MUS112' data the 0th item returned rather than 'CS134' as in the previous question? _____

4. Examine the following Python code:

```
>>> mixed = ['P', 'd', 5, 16, 2018]  
>>> sorted(mixed)  
TypeError: '<' not supported between instances of 'int' and 'str'
```

a. What is/are the types of the elements in mixed?

b. Why might the call to `sorted(mixed)` be throwing an error?



c. Below is a partially completed function we'd like to call with `sorted(mixed, key=return_ord_value)` so that the mixed list can be sorted:

```
def return_ord_value(element):  
    ''' Returns the ASCII value for an element if it is a  
        character, otherwise assumes that the given element is a  
        number and returns the number itself '''  
    if type(element) == str:  
        # (i) should return the ASCII value!  
        # (ii) How & when to return the number itself?
```

What does the `element` parameter represent in `return_ord_value`? _____

What might the `if type(element) == str` code be doing?

Write a line of code that replaces the (i) comment with what it should be doing:

Write a line of code that replaces the (ii) comment with what it should be doing:

Application Questions: Use the Python Interpreter to check your work

1. Write some code that sorts a list of tuples by age in a given year. The list, `people`, will have elements in the following format: `(name_str, birth_year, given_year, is_happy_bool)`

Hint: Write a function, `age_in(seq)`, that takes as an argument a sequence, `seq`, where the 1th element is the birth year, and the 2th element is a given year.

2. Write some code that when the following line of code is entered, will return the list sorted in ascending absolute value order: `sorted([-50, 50, -29, 27, 9], key=abs_value)`

Hint: The above call should return `[8, 27, -29, -50, 50]`

3. Write a function and a call to `sorted(...)` that will sort a mixed list of strings and integers appropriately. Here is an example list: `["Pixel", "dog", 5, 16, 2018]`
