PROBLEM 1:  (What will Python display? (24 points) (Estimate: 8 minutes))

For each code snippet, determine the output next to its corresponding `print` statement. If an error is produced, write "Error". Example outputs are provided for the initial statements.

OUTPUT:

```python
seta = set([6,3,2])
print(seta) # 1 {6, 3, 2}
print(list(seta)) # 2 [6, 3, 2]
print(sorted(seta)) # 3 [2, 3, 6]
print(tuple(seta)) # 4 (2, 3, 6)
```

Part A (4 points)

```python
nums = (4, 2, 3, 2, 4)
print(len(nums)) # 1
print(nums[1]) # 2
newNums = set(nums)
print(len(newNums)) # 3
print(newNums[1]) # 4
```

Part B (4 points)

```python
seta = {"A", "b", "c", 1, 2, 3}
setb = {"A", "B", "Z", 1, 2}
print(seta | setb) # 1
print(seta & setb) # 2
print(seta - setb) # 3
print(seta ^ setb) # 4
```

Part C (4 points)

```python
d = {"w": 24, "a": 18, "m": 45}
d["w"] = 10
d["z"] = 30
print(18 in d) # 1
print(d["w"]) # 2
print(d.items()) # 3
print(sorted(d.values())) # 4
```
PROBLEM 2:  (List Comprehensions (16 pts) (Estimate: 10 minutes))

Write a list comprehension to solve the given problems. Your solutions should be flexible enough to work with different inputs. That means if we changed the values inside the lists, sets, tuples, or dictionaries given, your code would still calculate the correct answer. Assume strings are lowercase and sequences have at least one item.

Each of these must be written in one line (using a list comprehension) for full credit. You can still get partial credit if you can solve the problem in more than one line (even if no list comprehension is used).

Here is an example:

The variable result should calculate the list of words from vehicles that have the letter "a" in their word. Using vehicles below, write the list comprehension that would store the following list into the variable result: ["train", "airplane", "car", "longboard"].

vehicles = ["train", "airplane", "car", "truck", "bike", "longboard"]

ANSWER:
result = [w for w in vehicles if "a" in w]

Part A (4 points)
The variable result should calculate the list of words of length 4 that appear in both lst1 and lst2. Using lst1 and lst2 below, write the list comprehension that would store the following list into the variable result: ["magic", "genie"]. The order of the words does not matter.

lst1 = ["unicorn", "magic", "dragons", "genie"]
lst2 = ["genie", "dragons", "rainbow", "magic"]

result =

Part B (4 points)
The variable result should calculate the list of strings from words (shown below) that contain non-repeated letters. Using words below, write the list comprehension that would store the following list into the variable result: ["sprint", "boxing"]. The order of the words does not matter.

words = ["sprint", "tennis", "volleyball", "boxing"]

result =
Part C (4 points)
The variable result should calculate the list of letters from word (shown below) that are non-unique letters. Using word below, write the list comprehension that would store the following list into the variable result: ["o", "o", "k", "k", "e", "e", "e"]. The order of the letters does not matter.

word = "bookkeeper"

result =

Part D (4 points)
The variable result should calculate the sorted (in increasing order) list of the unique values in the dictionary d that are greater than or equal to 10. Using the dictionary d below, write the list comprehension that would store the following list into the variable result: [10, 30, 40].

d = {"wizard": 10, "orc": 40, "elf": 5, "knight": 30, "hobbit": 2}

result =
PROBLEM 3 :  (Spot the Bug (15 points) (Estimate: 15 minutes))

Consider the following code snippet that contains an error:

```python
01 def findFirstEven(nums):
02     i = 0
03     while nums[i] % 2 != 0 and i < len(nums):
04         print("Checking number at index:", i)
05         i += 1
06     print("First even number found at index:", i)
07
08     if __name__ == "__main__":
09         numbers = [1, 9, 5, 7]
10         findFirstEven(numbers)
```

This function intends to find and print the index of the first even number in a list. If there are no even numbers in the list, then it will print "There are no even numbers." But this function is buggy! The expected output (assuming the code worked correctly) is:

```
Checking number at index: 0
Checking number at index: 1
Checking number at index: 2
Checking number at index: 3
There are no even numbers.
```

Answer the following questions regarding the above code.

a) What line number contains the function call?

b) What line number contains the function definition header?

c) What is the name of the argument?

d) What is the name of the parameter?
e) If the buggy code were to execute, what would be the value of \( i \) when the first iteration of the loop finished executing? Now, assume all the bugs were removed and the code worked as intended, what would be the value of \( i \) when the first iteration of the loop with the corrected code finished executing?

Actual value of \( i \) after the first iteration (using the buggy version):

Expected value of \( i \) after the first iteration (if bug(s) were removed):

f) If the buggy code were to execute, what would be the value of \( i \) when the last iteration of the loop finished executing? That is, the last loop iteration that is able to successfully finish iterating before crashing. Now, assume all the bugs were removed and the code worked as intended, what would be the value of \( i \) when the last iteration of the loop with the corrected code finished executing?

Actual value of \( i \) after the last iteration (using the buggy version):

Expected value of \( i \) after the last iteration (if bug(s) were removed):

g) What is the actual output (if any) when this buggy program is executed? If nothing is displayed, write "nothing". If an error is displayed, name or describe the error.

h) What line(s) of code have error(s)? There may be more than one line of code with an error.
i) Using the space below, rewrite the function with the error(s) removed.
PROBLEM 4 :  (Seating Chart (12 points) (Estimated time: 7 minutes))

A small theater has a seating arrangement organized in a grid pattern, where each seat is referenced by its row and column numbers. Write the function `findSeat` to check if a specific seat is available. The function should accept two arguments: a list of lists named `seatingChart`, where each inner list represents a row of seats, and a tuple named `seat` representing the desired seat coordinates (row, column). Each seat can either be available, occupied, or not even exist (out of bounds). The function should return True if the seat is available and False otherwise.

The indices for rows and columns start at 0. Be sure to check that the seat’s coordinates are within the bounds of the seating chart to avoid index errors. If it’s not, return False.

Examples of function calls and expected returns:

<table>
<thead>
<tr>
<th>Function call</th>
<th>Return value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>findSeat([[&quot;O&quot;], [&quot;O&quot;], [&quot;O&quot;], [&quot;A&quot;]], (3, 0))</td>
<td>True</td>
<td>This theater has 4 rows of seats such that each row contains 1 seat each. (3,0) means we are searching for the seat in row position 3, column 0. Since that seat is assigned the value &quot;A&quot;, it is Available, so return True.</td>
</tr>
<tr>
<td>findSeat([[&quot;A&quot;, &quot;O&quot;], [&quot;A&quot;, &quot;A&quot;], [&quot;O&quot;, &quot;A&quot;]], (0, 1))</td>
<td>False</td>
<td>This theater has 3 rows of seats such that each row contains 2 seats each. (0,1) means we are searching for the seat in row position 0, column 1. Since that seat is assigned the value &quot;O&quot;, it is Occupied, so return False.</td>
</tr>
<tr>
<td>findSeat([[&quot;A&quot;, &quot;O&quot;], [&quot;A&quot;, &quot;A&quot;], [&quot;O&quot;, &quot;A&quot;]], (0, 3))</td>
<td>False</td>
<td>This theater has 3 rows of seats such that each row contains 2 seats each. (1,3) means we are searching for the seat in row position 1, column 3. Since there is no column 3, this seat does not exist, so return False.</td>
</tr>
<tr>
<td>findSeat([[&quot;A&quot;, &quot;O&quot;, &quot;A&quot;, &quot;A&quot;], [&quot;A&quot;, &quot;A&quot;, &quot;O&quot;, &quot;O&quot;], [&quot;A&quot;, &quot;O&quot;, &quot;A&quot;, &quot;O&quot;], [&quot;A&quot;, &quot;A&quot;, &quot;A&quot;, &quot;O&quot;]], (2, 2))</td>
<td>True</td>
<td>This theater has 4 rows of seats such that each row contains 4 seats each. (2,2) means we are searching for the seat in row position 2, column 2. Since that seat is assigned the value &quot;A&quot;, it is available, so return True.</td>
</tr>
</tbody>
</table>

Complete the function on the next page.
def findSeat(seatingChart, seat):
PROBLEM 5: (Words Collector (15 points) (Estimate: 10 minutes))

In an effort to build a unique word collection, you are tasked with writing a function named `uniqueWordsCollector`, which takes a list of strings named `lines` and an integer named `minSize`. This function should analyze `lines`, where each string contains multiple words separated by spaces. The goal is to extract all unique words that have at least `minSize` characters and return them as a set. Assume all characters are lowercase and no punctuation. The order of the returned values does not matter.

Examples of function calls and expected returns:

<table>
<thead>
<tr>
<th>Function call</th>
<th>Return value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uniqueWordsCollector([&quot;quick brown fox&quot;, &quot;sleepy brown dog&quot;], 5)</code></td>
<td>{&quot;sleepy&quot;, &quot;quick&quot;, &quot;brown&quot;}</td>
<td>The following strings consist of a length of at least 5 characters: &quot;sleepy&quot;, &quot;quick&quot;, and &quot;brown&quot;. The following strings contain less than 5 characters: &quot;fox&quot; and &quot;dog&quot;. The string &quot;brown&quot; is repeated, so we only keep one instance of it.</td>
</tr>
<tr>
<td><code>uniqueWordsCollector([&quot;hello world&quot;, &quot;to be or not to be&quot;, &quot;you got this&quot;], 3)</code></td>
<td>{&quot;hello&quot;, &quot;this&quot;, &quot;got&quot;, &quot;not&quot;, &quot;world&quot;, &quot;you&quot;}</td>
<td>The following strings consist of a length of at least 3 characters: &quot;hello&quot;, &quot;this&quot;, &quot;got&quot;, &quot;not&quot;, &quot;world&quot;, and &quot;you&quot;. The following strings contain less than 3 characters: &quot;to&quot;, &quot;be&quot;, and &quot;or&quot;.</td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def uniqueWordsCollector(lines, minSize):
```
PROBLEM 6:  (Baking Contest (17 points) (Estimated time: 15 minutes))

In a baking contest, each item is scored solely on taste by judges. Write the function `bestBaker` to determine the winner. The function takes a list of tuples named `scores`, each tuple representing a contestant’s taste score given by a judge in the format: `(name, taste_score)`. The total score for a contestant is the sum of all their taste scores, which are integers. The function should return the name of the contestant with the highest total taste score. Assume there will be no ties.

Examples of function calls and expected returns:

<table>
<thead>
<tr>
<th>Function call</th>
<th>Return value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bestBaker([(&quot;Charles&quot;,10), (&quot;Conor&quot;,8), (&quot;Charles&quot;,10), (&quot;Conor&quot;,5)])</code></td>
<td>&quot;Charles&quot;</td>
<td>Charles has a total taste score of 20, while Conor’s is 13. So Charles wins.</td>
</tr>
<tr>
<td><code>bestBaker([(&quot;Abigail&quot;,2), (&quot;Elisabeth&quot;,4), (&quot;Ayda&quot;,2)])</code></td>
<td>&quot;Elisabeth&quot;</td>
<td>The total taste score for Abigail is 2, Elisabeth is 4, and Ayda is 2. This means Elisabeth won with the highest combined score.</td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def bestBaker(scores):
```
Python Reference Sheet for Compsci 101, Exam 2, Fall 2023

On this page we'll keep track of the Python types, functions, and operators that we've covered in class. You can also review the online Python References for more complete coverage, BUT NOTE there is way more python in the there then we will cover! The reference page below is all you should need to complete the exam and more.

<table>
<thead>
<tr>
<th>Mathematical Operators</th>
<th>Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>addition</td>
<td>4 + 5 = 9</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>subtraction</td>
<td>9 - 5 = 4</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>multiplication</td>
<td>3*5 = 15</td>
</tr>
<tr>
<td></td>
<td>/ and //</td>
<td>division</td>
<td>6/3 = 2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6/4 = 1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6//4 = 1</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>mod/remainder</td>
<td>5 % 3 = 2</td>
</tr>
<tr>
<td></td>
<td>**</td>
<td>exponentiation</td>
<td>3<strong>2 = 9, 2</strong>3 = 8</td>
</tr>
</tbody>
</table>

| String Operators      | +      | concatenation | "ab"+"cd"="abcd" |
|                      | *      | repeat        | "xo"*3 = "xoxoxo" |

| Comparison Operators  | ==     | is equal to   | 3 == 3 is True  |
|                      | !=     | is not equal to | 3 != 3 is False |
|                      | >=     | is greater than or equal to | 4 >= 3 is True |
|                      | <=     | is less than or equal to  | 4 <= 3 is False |
|                      | >      | is strictly greater than | 4 > 3 is True |
|                      | <      | is strictly less than  | 3 < 3 is False |

| Boolean Operators     | x=5    |              |              |
|                      | not    | flips/negates the value of a bool | (not x == 5) is False |
|                      | and    | returns True only if both parts of it are True | (x > 3 and x < 7) is True |
|                      |        |              | (x > 3 and x > 7) is False |


Page 1 of 6
### Type Conversion Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>int(x)</td>
<td>turn x into an integer value</td>
<td>int(&quot;123&quot;) == 123</td>
</tr>
<tr>
<td></td>
<td>int(x)</td>
<td>int(5.8) == 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>int can fail, e.g., int(&quot;abc&quot;) raises an error</td>
</tr>
<tr>
<td>float(x)</td>
<td>turn x into an float value</td>
<td>float(&quot;2.46&quot;) == 2.46</td>
</tr>
<tr>
<td></td>
<td>float(x)</td>
<td>float can fail, e.g., float(&quot;abc&quot;) raises an error</td>
</tr>
<tr>
<td>str(x)</td>
<td>turn x into a string value</td>
<td>str(432) == &quot;432&quot;</td>
</tr>
<tr>
<td>type(x)</td>
<td>the type of x</td>
<td>type(1) == int</td>
</tr>
<tr>
<td></td>
<td>type(x)</td>
<td>type(1.2) == float</td>
</tr>
</tbody>
</table>

### String Index and Splicing

**s = "colorful"**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>s[x]</td>
<td>index a character</td>
<td>s[0] == 'c'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s[-3] == 'f'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s[5] == 'f'</td>
</tr>
<tr>
<td>s[x:y]</td>
<td>splice of string, substring from index x up to but not including index y</td>
<td>s[2:5] == 'lor'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s[:5] == 'color'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s[4:-1] == 'rfu'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s[5:] == 'ful'</td>
</tr>
</tbody>
</table>

### String Functions

**s = "colorful"**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>.find(str)</td>
<td>index of first occurrence</td>
<td>s.find(&quot;o&quot;) == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s.find(&quot;e&quot;) == -1</td>
</tr>
<tr>
<td>.rfind(str)</td>
<td>index of last occurrence</td>
<td>s.rfind(&quot;o&quot;) == 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s.rfind(&quot;e&quot;) == -1</td>
</tr>
<tr>
<td>.index(str)</td>
<td>same as .find(str), error if str not in string</td>
<td>s.index(&quot;o&quot;) == 1</td>
</tr>
<tr>
<td>.count(str)</td>
<td>number of occurrences</td>
<td>s.count(&quot;o&quot;) == 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s.count(&quot;r&quot;) == 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s.count(&quot;e&quot;) == 0</td>
</tr>
<tr>
<td>.strip()</td>
<td>copy with leading/trailing whitespace removed</td>
<td>&quot; big &quot;.strip() == &quot;big&quot;</td>
</tr>
</tbody>
</table>
### split()
- **list of "words" in s**
- **"big bad dog".split()**
  - `== ["big", "bad", "dog"]`

### .split("")
- **list of "items " in s that are separated by a comma**
- **In general can split on any string, not just a comma, e.g., s.split(".") will split on a colon and s.split("gat") will split on the string "gat".**
- **"this, old, man".split(",")**
  - `== ["this", "old", "man"]`

### ''.join(lst)
- **concatenate elements of lst, a list of strings, separated by "," or any string**
- **".join(['a','b','c'])**
  - `== "a:b:c"`

### .startswith(str)
- **boolean if starts with string**
- **s.startswith("color")**
  - `== True`
- **s.startswith("cool")**
  - `== False`

### .endswith(str)
- **boolean if ends with string**
- **s.endswith("ful")**
  - `== True`
- **s.endswith("color")**
  - `== False`

### .upper()
- **uppercase of s**
- **s.upper()**
  - `== "COLORFUL"`

### .lower()
- **lowercase of s**
- **"HELLO".lower()**
  - `== "hello"`

### .isupper()
- **boolean is uppercase**
- **'A'.isupper()**
  - `== True`
- **'a'.isupper()**
  - `== False`

### .islower()
- **boolean is lowercase**
- **'A'.islower()**
  - `== False`
- **'a'.islower()**
  - `== True`

### .isalpha()
- **boolean is alphabetic character**
- **'3'.isalpha()**
  - `== False`
- **'?'.isalpha()**
  - `== False`
- **'z'.isalpha()**
  - `== True`

### .capitalize()
- **capitalized s**
- **s.capitalize()**
  - `== "Colorful"`

### .replace(str1, str2)
- **replace all occurrences of str1 with str2**
- **s.replace('o','y')**
  - `== "cylyrful"

### .replace(str1, str2,n)
- **replace the first n occurrences of str1 with str2**
- **s.replace('o','y',1)**
  - `== "cylorful"

### Miscellaneous Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help(x)</td>
<td>documentation for module x</td>
</tr>
<tr>
<td>len(x)</td>
<td>length of sequence x, e.g., String or List</td>
</tr>
<tr>
<td>list(str)</td>
<td>a list of the characters from string str</td>
</tr>
</tbody>
</table>

```
len("duke") == 4
list("cards") == ['c', 'a', 'r', 'd', 's']
```
| **sorted(x)** | return list that is sorted version of sequence/iterable x, doesn't change x | sorted("cat") == ['a', 'c', 't'] |
| **range(x)** | a list of integers starting at 0 and going up to but not including x | range(5) == [0, 1, 2, 3, 4] |
| **range(start, stop)** | a list of integers starting at start and going up to but not including stop | range(3, 7) == [3, 4, 5, 6] |
| **range(start, stop, inc)** | a list of integers starting at start and going up to but not including stop with increment inc | range(3, 9, 2) == [3, 5, 7] |
| **min(x, y, z)** | minimum value of all arguments | min(3, 1, 2) == 1 |
| | | min("z", "b", "a") == "a" |
| **max(x, y, z)** | maximum value of all arguments | max(3, 1, 2) == 3 |
| | | max("z", "b", "a") == "z" |
| **abs(x)** | absolute value of the int or float x | abs(-33) == 33 |
| | | abs(-33,5) == 33,5 |

### List index, splicing and concatenation

**lst** = [3, 6, 8, 1, 7]

| lst[x] | index an element | lst[0] == 3 |
| lst[-1] == 7 |
| lst[x:y] | splice of list, sublist from index x up to but not including index y | lst[1:3] == [6, 8] |
| lst[:4] == [3, 6, 8, 1] |
| lst[3:] == [1, 7] |
| + operator | concatenate two lists | [3, 4] + [1, 3, 2] == [3, 4, 1, 3, 2] |

### List Functions

**lst** = [3, 6, 8, 1, 7]

| sum(lst) | returns sum of elements in list lst | sum([1, 2, 4]) == 7 |
| max(lst) | returns maximal element in lst | max([5, 3, 1, 7, 2]) == 7 |
| lst.append(...) | append an element to lst, changing lst | [1, 2, 3].append(8) == [1, 2, 3, 8] |
| lst.insert(pos, elt) | append elt to lst at position pos, changing lst | [1, 2, 3].insert(1, 8) == [1, 8, 2, 3] |
| lst.extend(lst2) | append every element of lst2 to lst | [1, 2, 3].extend([8, 9]) == [1, 2, 3, 8, 9] |
```plaintext
<table>
<thead>
<tr>
<th>lst.remove(elt)</th>
<th>remove first occurrence of elt from lst</th>
<th>[1, 2, 3, 2, 3, 2].remove(2) == [1, 3, 2, 3, 2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>lst.sort()</td>
<td>sorts the elements of lst</td>
<td>lst = [3, 6, 8, 1, 7]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lst.sort()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lst is now [1, 3, 6, 7, 8]</td>
</tr>
<tr>
<td>lst.index(elt)</td>
<td>return index of elt in lst, error if elt not in lst</td>
<td>[1, 5, 3, 8].index(5) == 1</td>
</tr>
<tr>
<td>lst.count(elt)</td>
<td>return number of occurrences of elt in lst</td>
<td>[1, 2, 1, 2, 3].count(1) == 2</td>
</tr>
<tr>
<td>lst.pop()</td>
<td>remove and return last element in lst, so has side-effect of altering list and returns value.</td>
<td>lst = [3, 6, 8, 1, 7]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x = lst.pop()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x is 7, lst is [3, 6, 8, 1]</td>
</tr>
<tr>
<td>lst.pop(index)</td>
<td>remove and return element at position index in lst, so has side-effect of altering list and returns value. Default index is last value.</td>
<td>lst = [3, 6, 8, 1, 7]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x = lst.pop(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x is 6, lst is [3, 8, 1, 7]</td>
</tr>
</tbody>
</table>

**Math Functions (import math)**

- **math.pi**
  - 3.1415926535897931

- **math.sqrt(num)**
  - returns square root of num as float
  - math.sqrt(9) == 3.0

**File Functions**

- **open("filename")**
  - opens a file, returns file object
  - f = open("foo.txt")

- **open("filename", mode)**
  - specify mode of 'r', 'a', 'w', return file object
  - f = open("foo.txt", "a")

- **f.read()**
  - returns the entire file as one string
  - s = f.read()

**Random Functions (import random)**

- **random.choice(list_ofChoices)**
  - returns a random element from list_of_choices. Gives an error if list_of_choices has length 0.

- **random.randint(start, end)**
  - Returns a random integer between start and end. Unlike range() and list slicing, the largest value it can return is end, not end-1.

- **random.random()**
  - Returns a random float between 0 and 1.

**Set Functions**

- **set(lst)**
  - returns a set of the elements from list lst

- **s.add(item)**
  - adds the item into the set, and returns nothing.

- **s.update(lst)**
  - adds the elements in the list lst into the set, and returns nothing.

- **s.remove(item)**
  - removes the item from the set, error if item not there.

- **s.union(t)**
  - returns new set representing s UNION t, i.e., all elements in either s OR t, t can be any iterable (e.g., a list)```
s.intersection(t) returns new set representing s INTERSECT t, i.e., only elements in both s AND t, t can be any iterable (e.g., a list)

s.difference(t) returns new set representing s difference t, i.e., elements in s that are not in t

s.symmetric_difference(t) returns new set representing elements in s or t, but not in both

s | t returns/evaluates to union of s and t, both must be sets.

s & t returns/evaluates to intersection of s and t, both must be sets.

s - t returns/evaluates to set with all elements in s that are not in t

s ^ t returns/evaluates to set with all elements from s and t that are not in both s and t

Dictionary Functions
d[key] returns the value associated with key, error if key not in dictionary d

d.get(key) returns value associated with key, returns None if key not in dictionary d

d.get(key,default) returns value associated with key, returns default if key not in d

d.keys() returns a list/view of the keys in dictionary

d.values() returns a list/view of values in dictionary

d.items() returns a list/view of tuples, (key,item) pairs from dictionary

d.update(dict) updates the dictionary with another dictionary dict

Lambda Functions

```python
lst = [('c', [4, 2, 8]), ('h', [2, 7, 1, 6]), ('b', [3, 9])]

f = lambda x: len(x[1])
y = sorted(lst, key=f)
y is [( 'b', [3, 9]), ( 'c', [4, 2, 8]), ( 'h', [2, 7, 1, 6])] as it sorts tuples on the length of the lists in the index 1 position

y = sorted(lst, key= lambda x: x[1])
y is [( 'h', [2, 7, 1, 6]), ( 'b', [3, 9]), ( 'c', [4, 2, 8])] as it sorts tuples based on the first element in each list (the 2, 3, and 4)
```

Image Library Functions

```python
Image.open(fname) opens and returns image
im.show() displays image im
im.getdata() returns generator of all pixels in im
im.putdata(pixlist) modifies image by setting all pixels to pixlist
im.size returns tuple that is (width,height) of image
Image.new('RGB', size) creates and returns a new image with dimensions of tuple size
```