Text: How to think like a computer scientist (learning with Python[1]).

Reading Assignment 7: Chapter 7 (Strings).

Questions: Would you call integer values `atomic`[2]? Would you say that these values are in some sense indivisible (that you can't take them apart)? How else could a value be? What does the book mean by `compound type`? What benefit is associated with such a type? What is the difference between the integer `123` and the string `"123"`? Don't we hit the keyboard three times to type `123` (the number) just like we do for the string? What's `raw_input`'s return value? Have you ever seen a number? Have you ever seen a string of characters? How do you convert a string to an integer? How do you obtain the last digit in a number? How do you get the last character in a string[3]? How do you obtain the first character in a string? How do you obtain the first digit in a number? Is there any such thing as an empty string? Is it a string? Can you convert it to a number? Can you use it in an if statement, or in a loop as the condition? Explain your answers.

If `a = "blueberry"` what does `a[2]`, `a[1]`, `a[0]` evaluate to?

Same questions for `a[-1]` and `a[-2]`. Any surprises?
What is an `index`? What is `a[2-1]`? What is `a[len(a)-1]`?

What is `a[len(a)]`? What does `a[-len(a)]` evaluate to?

Same questions for `a[-len(a)-1]` and `a[-len(a)+1]`.

Define what we mean by `traversal` of a string. Write a function that receives a word (as a string, of course) and prints back the word without the vowels, like this:

```
>>> steno("nectarine")
'nctrn'
>>> steno("blueberry")
'blbrry'
>>>
```

Change then the code to surround the vowels with parens instead of not showing them.

```
>>> steno("nectarine")
'n(e)ct(a)r(i)n(e)'
>>> steno("blueberry")
'bl(u)(e)b(e)rry'
>>>
```

---

[1] `http://www.cs.indiana.edu/classes/a202-dger/fall2006/a201-text/thinkCSpy.pdf`
[2] Comes from Greek *atomos* which means *indivisible*.
[3] How do you calculate the *length* of a string?

Use a `for … in …` loop to produce the reverse[4] of a given string.

Solve/finish the exercise listed at the bottom of page 74.

What is a string `slice`? If `a = "blueberry"` evaluate the following:

- `a[2:3]`
- `a[2:]`
- `a[:3]`
- `a[:]`
- `a[-1:-3]`
- `a[:-1]`
- `a[1:1]`

Explain how the comparison operators `==`, `<`, `<=`, `>`, `>=` and `!=` work with strings[5].

What does the following code fragment produce and why:

```
a = "readEval"
b = "read_eval"
c = "readeval"

print a < b, a < c, b < c
```

What does it mean for strings to be `immutable`?

Write code for a function (call it `fun`) that receives a non-empty word and replaces the first letter in it with an underscore, like this:

```
>>> fun("blueberry")
'_lueberry'
>>> fun("glue")
'_lue'
>>>
```

Is that how you implemented your `steno` function from the previous exercises?

Write the code for the modified `find` function mentioned in the exercise on page 76.

Write a function `count` which gets two arguments, a string and a character, and reports the number of times (perhaps zero) that the character appears in the string. Write another function `contains` which takes a string and a character and determines if the character appears in the string or not. Can you define `contains` in terms of `count`?

What is the benefit and purpose of the string module?

---

[4] That is, obtain and print `a[-1] + a[-2] + … + a[0]` where `a` is the given string.
[5] Discuss if they work any different for numbers.

Briefly describe each of the following:

1. `string.find(…)`
2. `string.lowercase`
3. `string.uppercase`
4. `string.digits`
5. `string.whitespace`

What's the difference between the `in` operator as presented in this section (7.10) and the one seen in the `for … in …` loop? Are there any similarities, too?

Where can you find the Python Library Reference[6]?

Programs to write:

- write a program that plays Hangman with the user
- write a program that shuffles[7] the letters in a word (word jumble)
- write a program to produce circular permutations of a given word[8]

Reading Assignment 8: Chapter 8 (Lists).

Questions: A ____ is an ordered set of values, where each value is identified by an index.

Do strings match the definition you just read? How do they match it and how don't they?

Both lists and strings are called _____ .

Is a string the same as a list of characters?

Can we have a list of strings?

What is a nested list? Can we have nested strings?

What values can we have in a list? Do they all have to be the same type[9]?

What does `range(...)` do[10]?

How do you create the following lists:

- `[4, 5, 6]`
- `[-2, 1, 3]`

---

[6] List the URL, like so: `http://docs.python.org/lib/`
[7] Randomly. For example with input python it could produce poynth
[8] Give it the word apple and it produces: pplea, pleap, leapp, eappl, and apple.
[9] Can you have an integer, a string, a list of integers and a list of strings in a list?
[10] Include all variants, with one, two and three arguments in your answer.

- `[-9, -8, -7, -6, -5]`
- `[-9, -10, -11, -12]`
- `[0, 1, 2]`

Is the empty string an empty sequence? Can lists be empty?

Write a function that receives a list and a value and counts the number of occurrences of the value in the given list. Provide two implementations: recursive and iterative.

If `a = [5, 4, 3, 2, 1, 0]` evaluate the following expressions:

- `a[0]`
- `a[-1]`
- `a[a[0]]`
- `a[a[-1]]`
- `a[a[a[a[2]+1]]]`

Can you change an element of a sequence? What if the sequence is a string?

Consider the following code: what is the value of a at the end?

```
a = [1, 2, 3]
a[2]=0
a[a[2]] = 5
a[1:2] = []
```

What is the difference between `a[1:2] = []` and `a[1] = []`? How would `a` change if we make this change in the last statement of the fragment presented?

What does `a + b` amounts to if `a` and `b` are lists?

Write a function that receives a list of integers and returns their sum[11].

How do you determine the length of a list?

If `a = [1, 2, [3, 4], 5]` what is the length of `a`? How many elements does it contain? How many integers? What does `len(a)` return?

Same questions after we make `a[1:2] = [[2, 3], 4]`.

What does `a[len(a)]` evaluate to[12]?

Write a function that receives a list (that could contain nested lists) of integers and reports the total the number of integers at all levels[13].

---

[11] What is list traversal?
[12] Assume `a` is a list: how do you obtain its last element?
[13] For example [1, [2, 3, 5], 4] is a list of three elements with five integers inside.

If `a` is `[1, 2, 3]` is there a difference between `not 1 in a` and `1 not in a`?

Is every `for` loop expressible through an equivalent `while` loop? Why or why not, and what is your justification? Is the reverse true?

Write loops that produce the first 20 terms in the sequences:

- `1, -1, 1, -1, 1, -1...`
- `1, 3, 5, 7, 9, 11, 13, ...`
- `1, 0, 2, -1, 3, -4, 5, -6, ...`

Is one type of loop usually more useful or convenient than the other?

Which one do you prefer? Why?

If `a` is `[1, 2, 3]`

- what is the difference (if any) between `a * 3` and `[a, a, a]`?
- is `a * 3` equivalent to `a + a + a`?
- what is the meaning of `a[1:1] = 9`?
- what's the difference between `a[1:2] = 4` and `a[1:1] = 4`?

What's `a[1:1]` if a is a string of at least two characters[14]?

What's the purpose of the `del` operator? Can you delete a slice?
Why is there so much space left on page 88?

What is a state diagram and where have we used it in this text?

What is the following Python interaction trying to communicate:

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> id(a)
10954592
>>> id(b)
12948944
>>> a[0] = 3
>>> a
[3, 2, 3]
>>> b
[1, 2, 3]
>>> id(a)
10954592
>>> id(b)
12948944
>>>
```

[14] What if the string is shorter?

Same question for this one:

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> id(a)
12012840
>>> id(b)
12773376
>>> b = a
>>> id(b)
12012840
>>> a[0] = 3
>>> a
[3, 2, 3]
>>> b
[3, 2, 3]
>>>
```

Is there any difference between the two?

Now explain this:

```
>>> a = 1
>>> b = 1
>>> id(a)
10901272
>>> id(b)
10901272
>>> b = 2
>>> id(b)
10901260
>>> a
1
>>>
```

Do you find anything related in this last example:

```
>>> a = [1, 2, 3]
>>> id(a)
10954592
>>> a[0] = 3
>>> id(a)
10954592
>>>
```

Why is `id(a) == id(b)` while `id(x) != id(y)` in the example below:

```
>>> a = 9
>>> b = 9
>>> id(a)
10901176
>>> id (b)
10901176
>>> id(a) == id(b)
```

```
True
>>> x = [1, 2, 3]
>>> y = [1, 2, 3]
>>> id(x) != id(y)
True
>>> id(x)
11428848
>>> id(y)
12943768
>>>
```

What is `aliasing`? The book shows that aliasing sometimes does not happen for strings? Why do you think that is so? Comment on the code below:

```
>>> a = "banana"
>>> b = "bana" + " na"
>>> id(a)
12929248
>>> id(b)
12930912
>>> c = "banana"
>>> id(c)
12929248
>>>
```

What's the simplest way to clone a list? What's `cloning`?

Write a function that sorts an array of integers in ascending order.

Write a function that reverses an array of integers (in situ, or as a clone).
Write a function that receives two lists and creates a third, that contains all elements of the first followed by all elements of the second.

Using nested lists write a function that implements matrix addition.

Does the slice operator always produces a new list?

Write a function to generate a matrix of given size (rows and columns) full of random values (positive integers between -50 and 50). Work out a show function that receives a matrix and prints it nicely, with all the values properly aligned, like this:

```
>>> show([[1, 2, 3], [101, 102, 19], [1, 12, 123]])
  1   2   3
101 102  19
  1  12 123
>>> show([[12, 201, 3], [1, 19], [123]])
 12 201   3
  1  19
123
>>>
```

Note the function works even if the argument is not a matrix proper.

Write a program that reverses the order of words in a sentence[15].

Write a program that reverses the order of the letters in the words in a sentence, leaving the order of the words in the sentence unchanged.

Combine the two programs above into one that reverses the order of the words in a sentence and the order of the letters in each word:

```
Echo> these are the days of miracle and wonder
these are the days of miracle and wonder
rednow dna elcarim fo syad eht era eseht
eseht era eht syad fo elcarim dna rednow
wonder and miracle of days the are these
Echo>
```

Answer the question on page 94.

Write a program that reads two times in military format (0900, 1730) and prints the number of hours and minutes between the two times. Here is a sample run. User input is in color.

```
Please enter the first time: 0900
Please enter the second time: 1730
8 hours 30 minutes
```

Extra credit if you can deal with the case that the first time is later than the second time:

```
Please enter the first time: 1730
Please enter the second time: 0900
15 hours 30 minutes
```

Write a program that reads a string (or receives one, if you write a function) and swaps the first with the last letter: for example, `pooh` would become `hoop`.

Reading Assignment 9: Chapter 9 (Tuples).

Questions: How do tuples resemble strings? How do they resemble lists? How do they resemble neither? Is it hard to create a tuple with a single element? Is it even possible? Are tuples sequences like lists and strings or not? Is slicing working for them in the same way? What does `+` mean for tuples?

What is the difference between

```
a, b = b - a, a - b
```

and

[15] Use split from the string module.

```
a = b - a
b = a - b
```

How can you swap the values of two variables in Python[16]?

Explain what the following code does:

```
def swap(x, y):
    x, y = y, x

a = 3
b = 5

swap(a, b)

print a, b
```

How do you generate random numbers in Python?

Simulate a dice in Python.

Write a program to play the game of craps.

Solve the exercises on page 98.

Explain this code:

```
>>> a = [[0] * 3] * 3
>>> a
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> a[0][1] = 4
>>> a
[[0, 4, 0], [0, 4, 0], [0, 4, 0]]
>>>
```

Is there anything wrong[17] with it?

Describe and implement a test for the uniform distribution of the Python pseudorandom number generator. Use 8 buckets. Explain every line of your code.

Solve the exercise mentioned at the bottom of page 101.

Write a function that calculates the average of a list of numbers.
Write a function that receives a size, and two numbers (`low` and `high`) and returns a list of random numbers in the given range[18].

---

[16] With and without using a third variable.
[17] Fix it if there's anything wrong with it. Explain why it's correct if there's nothing wrong with it.
[18] It's up to you what numbers (floating-point or integers) you generate.

Write another function that calculates the standard deviation in a list of numbers received as input: the standard deviation is the average of the squared deviations[19] to the mean.

Solve the exercise listed at the bottom of page 102.

Reading Assignment 10: Chapter 10 (Dictionaries).

Questions: What is a dictionary. Are dictionaries sequences? Is there any such thing as the empty dictionary? How can you add, delete, modify entries in a dictionary? In what circumstances are dictionaries useful? Define the terms `key`, and `key-value` pairs.

Is a dictionary suitable for storing username-password pairs? How would you do that using regular lists?

Look at the code below and explain what it illustrates:

```
>>> hash = { 'one' : 1, 'two' : 2, 'three' : 3 }
>>> hash
{'three': 3, 'two': 2, 'one': 1}
>>>
```

How could you list the entries in this dictionary in the order you want[20]?

Implement matrix multiplication using dictionaries.

Implement the following procedure to construct magic `n`-by-`n squares;`

- it works only if `n` is odd.
- Place a `1` in the middle of the bottom row.
- After `k` has been placed in the `(i, j)` square, place `k+1` into the square to the right and down, wrapping around the borders.
- However,
    1. if the square to the right and down has already been filled, or
    2. if you are in the lower right corner,

  then you must move to the square straight up instead.

Here's the `5`-by-`5` square that you get if you follow this method:

```
 11  18   25    2    9
 10  12   19   21    3
  4   6   13   20   22
 23   5    7   14   16
 17  24    1    8   15
```

[19] Individual deviations.
[20] Such as in the order: one, two three.

Check that the square above is magic.

Calculate the `3`-by-`3`, `7`-by-`7` and `13`-by-`13` `magic squares`.

How do you find how big a dictionary is?

What is the `keys` method doing and how do you invoke[21] it[22]?

How do you find out if a dictionary entry exists or not?

Is any dictionary an object? What do you think that means?

How do you modify a dictionary and keep a copy of the original?

Is the `get` method offering any break if the entry does not exist in the dictionary?

How can a dictionary be used to speed up the calculation of fibonacci numbers? If in the original solution we waste a lot of time and in the dictionary implementation we get some of that back, is there any tradeoff we make (like spending some other kind of resource, that is related to the use of dictionary, to not duplicate any calculations)? Is it possible that this tradeoff bring us back in the same starting situation?

What does it mean for a value to be global?

How long can a long number be? Can you calculate $2^{23456}$? How many digits does the answer have? How long does it take for the calculation to conclude? How does Python resort to long integers (does it detect the need automatically, do you need to use a special syntax like in the case of strings and floats, etc.)?

Write a program that determines the frequency of occurrence of letters in a string.

Write a program that determines the frequency of occurrence of vowels in a string.

How can you list the entries in a dictionary in alphabetical order?

Give examples of `extend`, `reverse` and `append` in a list context.

*Turn in a Word document with all the questions answered completely.*

---

[21] What is that? What do we mean by *invocation*?
[22] Same question about `values` and `items`.