Teach Python: Sample Exercises

This is a somewhat structured dump of (hopefully) relevant exercises from my handouts for Sheridan's *Programming Principles* course (the intro programming course for the Bachelor of Applied Computer Science – Mobile). Content covered includes basic I/O, variables, types, expressions, if, while, functions, and lists. There are solutions to some of these exercises in the accompanying folders.

A. Basic Input-Processing-Output Exercises

1. Write a simple program that asks the user for a product name and a price. Then output the total cost of the product with HST added, like this (don't worry about formatting for now).

   ```
   Product name?
   computer
   How much does a computer cost?
   500.00
   The total cost of computer including HST will be $ 565.0
   ```

2. Write a program that asks the user for two integers x and y and then outputs the result of raising x to the power y.

3. Write a program that asks the user separately for their first name and last name. Then ask for their favorite positive integer, and say hello to them that many times.

   First Name?
   Sam
   Last Name?
   Scott
   Favorite positive integer?
   5
   Hello Sam Scott! Hello Sam Scott! Hello Sam Scott! Hello Sam Scott! Hello Sam Scott!

4. Write a program that converts a temperature from Celsius to Fahrenheit using the formula $F = \frac{9}{5}C + 32$.

5. Write a program that asks the user to enter two points (x_1, y_1) and (x_2, y_2) and displays the distance between them using the formula $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. (Reminder: $\sqrt{x} = x^{0.5}$)

6. Change your answer to question 1 so that the dialog with the user looks like this:

   ```
   Product name? computer
   How much does a computer cost? 500.00
   The total cost of computer including HST will be $ 565.0
   ```

7. Try exercises 2.3 and 2.4 in *Think Python*.

8. Write a program that gets a temperature in Celsius (t_c) and the wind speed in kilometers per hour (v_k), then outputs the wind chill according to this formula:

   $$t_{wc} = 35.74 + 0.6215t_f - 35.75{v_m}^{0.16} + 0.4275t_f{v_m}^{0.16}$$

   In the formula above, t_f is the temperature in Fahrenheit and v_m is the wind speed in miles per hour. You'll have to convert the inputs before using this formula!

   Note that this formula is only valid for wind speeds of 2mph or greater and in the temperature range -58°F to 41°F. But for now we don't know how to enforce that…

B. Math and Built In Function Exercises

Exercises 1 and 3 adapted from *Introduction to Java Programming, 10th Edition* by Y. Daniel Lang.

1. The great circle distance is the distance between two points on the surface of a sphere. If you want to know the distance between two points on the surface of the earth, you can approximate it using the great circle distance.

   Let (x_1, y_1) and (x_2, y_2) be the geographical latitude and longitude of two points. The great circle distance between the two points can be computed using the following formula:

   $$d = radius \times \text{acos}(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2))$$

   Write a program that prompts the user to enter the latitude and longitude of two points on the earth in degrees and displays its great circle distance. The average earth radius is 6,371.01 km. Note that you need to convert the degrees into radians using the `math.radians()` function since the Python trigonometric methods use radians. The latitude and longitude degrees in the formula are for north and west. Use negative to indicate south and east degrees.

   You should round your answers to the nearest metre.

   Here is a sample run:

   ```
   Enter point 1 (latitude and longitude) in degrees: 39.55, -116.25
   Enter point 2 (latitude and longitude) in degrees: 41.5, 87.37
   The distance between the two points is 10691.792 km
   ```

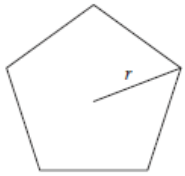
2. Write a program that computes the average of 3 grades entered by the user. The grades should be in the range 0 to 100 (inclusive). If the user enters a number out of that range, use the `max` and `min` functions to correct what they entered (less than 0 becomes 0, greater than 100 becomes 100). Round the answer to the nearest integer.

   ```
   Enter your three grades:
   50
   -50
   150
   Your grades are 50, 0 and 100
   ```

```
Your average is 50
```

3. Write a program that prompts the user to enter the length from the center of a pentagon to a vertex and computes the area of the pentagon, as shown in the following figure.

   The formula for computing the area of a pentagon is $Area = \frac{5s^2}{4\tan\frac{\pi}{5}}$, where s is the length of a side. The side can be computed using the formula $s = 2r\sin\frac{\pi}{5}$, where r is the length from the center of a pentagon to a vertex. Round to two decimal places.

   Here is a sample run:

```
Enter the length from the center to a vertex: 5.5
The area of the pentagon is 71.92
```

C. Function Definition Exercises

1. Write a `print_signature` function that pauses for 2 seconds, outputs a custom signature (i.e. your name plus some other text marking the program as your own) to the python shell, and then pauses for another 2 seconds. Then go back to one of your previous exercises and paste the function in. Call it at the beginning and the end of your program.

   (Note: You can import the `time` module within the function. If you do this, you can paste it into any program and you don't have to remember to also add the import. It's ok to import the same module more than once – the subsequent imports will be ignored.)

2. Write a pygame version of your signature. All the pygame initialization should happen inside the function and the window should be closed when the function is finished. Paste this into one of your previous exercises.

3. Try exercises 3.1 and 3.2 in section 3.6 of *Think Python*.

D. More Function Definition Exercises

1. Exercise 3.3 from *Think Python* (section 3.16)

2. Write and test a function that displays, formatted correctly, the tip amount given the amount of the check and the percentage tip. For example if you call `tip(100,20)` it should print:

```
A 20% tip on $100.00 would be $20.00
```

3. Write and test a function that asks the user to input the check total and the percentage tip they want, then calls the function you wrote in question 2 to display the tip.

4. Exercise 3.4 from *Think Python* (section 3.16)

8. (hard) Try exercise 3.5 from the text (section 3.16). Although it's not explicitly stated, the idea here is that you should write functions to draw parts of the grid and use `do_four()` and `do_twice()` from exercise 3.4 to call the functions and build up the grid piece by piece.

E. If Statement Exercises

Questions 1b, 4, 5, 7, 8, 9 adapted from Y. Daniel Liang, *Introduction to Java Programming, 10th Edition*.

1. (Modulus & If)
   a. Write a function that implements the change-making algorithm using this interface:

```
def change(amount):
    """Prints the number of coins of each type required to make
    the given amount of change
    amount: The amount to return, in cents. Must be an integer
    multiple of 5."""
```

   b. Modify your program so that it only displays the coins that need to be returned (i.e. skip the zeros) like this:

      "$4.50 can be made with 2 toonies, 2 quarters."

   c. (Harder) Modify your program again so that if it's a single coin, it prints the amount using a singular word, like this:

      "4.55 can be made with 2 toonies, 2 quarters, 1 nickel."

   d. (Even harder) Can you get the "and" so that it's in the right place in all cases?

      "$4.55 can be made with 2 toonies, 2 quarters and 1 nickel."

2. (Modulus & If)
   a. Write a function that accepts an integer parameter that represents the number of seconds that have passed so far today (since midnight). The function should print the current time using a 24 hour clock (format is hh:mm:ss). Use if statements to make sure you print leading zeroes (i.e. don't output 1:4:5 for 3845. It should be 01:04:05)

```
def print_time(seconds):
    """Prints the time using a 24 hour clock format hh:mm:ss.
    seconds: the number of seconds that have passed since
    midnight."""
```

   b. The `time` module (the one that contains the `sleep()` function) also contains a function called `time()` that returns the number of seconds since an event known as the "UNIX Epoch" (Midnight, January 1, 1970 UTC). Figuring out the date from this time stamp can be done but it's tricky. It's easier to figure out the time of day. Write a function that prints the current time of day (UTC), using the `print_time()` function you wrote in question 2. Can you modify this function so that it prints the correct time for your time zone?

3. (If statement) Do exercise 5.4 from *Think Python* (section 5.14). To make it a bit easier, assume that the user will sort the integer arguments in ascending order.

4. (If statement with strings) Write a function to help someone look up a word in a paper dictionary (they have a computer, but no internet connection, ok?) The function should take two parameters: the word they are looking for and the word at the top of the current page. Then tell them whether they should search in the pages before or after this one.

5. (Modulus and If) Write a function that takes a three-digit positive or negative integer and displays a message to indicate whether it is a palindrome number. A number is palindrome if it reads the same from right to left and from left to right. If the number has more than 3 digits, you should print an error message. (You can assume it's an integer).

7. (If statement, arithmetic expressions) The two roots of a quadratic equation $ax^2 + bx + c = 0$ can be obtained using the following formulae:

   $r_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$ and $r_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$

   $b^2 - 4ac$ is called the discriminant of the quadratic equation. If it is positive, the equation has two real roots. If it is zero, the equation has one root. If it is negative, the equation has no real roots. Write a function takes parameters for *a*, *b*, and *c* and displays the result based on the discriminant. If the discriminant is positive, display two roots. If the discriminant is 0, display one root. Otherwise, display "The equation has no real roots".

   Then write a function that prompts the user to enter 3 floats for a, b and c and the calls the function you wrote to display the roots.

   Here are some sample runs:

```
Enter a, b, c:
1.0
3
1
The equation has two roots -0.381966 and -2.61803

Enter a, b, c: 1
2.0
1
The equation has one root -1

Enter a, b, c:
1
2
3
The equation has no real roots
```

8. (Modulus and if) An ISBN-10 (International Standard Book Number) consists of 10 digits: d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}. The last digit, d_{10}, is a checksum, which is calculated from the other nine digits using the following formula:

   $(d_1 \times 1 + d_2 \times 2 + d_3 \times 3 + d_4 \times 4 + d_5 \times 5 + d_6 \times 6 + d_7 \times 7 + d_8 \times 8 + d_9 \times 9)\ \%\ 11$

If the checksum is 10, the last digit is denoted as X according to the ISBN-10 convention. Write a function that takes the first 9 digits as a single integer (not string) and displays the 10-digit ISBN (including leading zeros). Then write a function to get an input from the user and call the function. Here are sample runs:

```
Enter the first 9 digits of an ISBN as integer: 013601267
The ISBN-10 number is 0136012671
Enter the first 9 digits of an ISBN as integer: 013031997
The ISBN-10 number is 013031997X
```

9. (Arithmetic and If) Write a program that prompts the user to enter a point (x, y) and checks whether the point is within the circle centered at (0, 0) with radius 10. For example, (4, 5) is inside the circle and (9, 9) is outside the circle. Use Pygame to draw the circle and the point so you can check whether or not your program got it right.

F. Boolean Expressions, Chained and Nested Conditoinals

Exercises 1d, 1e, 2a, 2b, 2c, 2e, 2f and 3c were adapted from Y. Daniel Liang, *Introduction to Java Programming, 10th Edition*.

1. Boolean Expressions

a. Groovy Numbers. A number is "groovy" if it is divisible by 13 or if it is both even and greater than 10. Write a program that takes a number from the user, and then says whether it is a groovy number or not. You will have to combine `and`, `or` and maybe some brackets for this one.

```
Gimme a number
39
That number is groovy
Gimme a number
10
That number is not groovy :-(
```

b. Tire Pressure. To be correctly inflated, a car's tires should all have a pressure between 35 and 45 psi. Also, the front tires should be within 2 psi of each other, and the rear tires should be between 2 psi of each other. Write a program that asks the user for the pressure of each tire, and then tells them whether the inflation is good or bad. Check all the conditions with a single Boolean expression. You will have to use the and operator, subtraction, and maybe some brackets for this one.

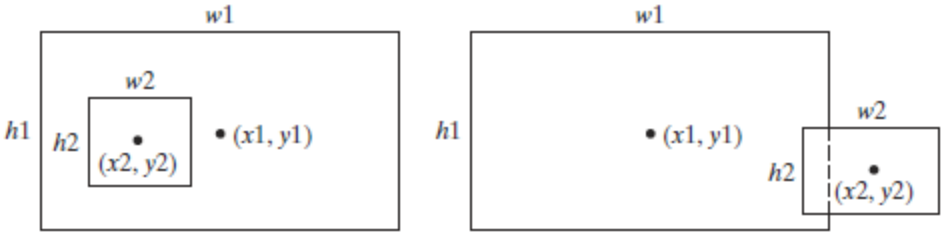
Make sure you test your program with lots of different situations (use what you know about branch and boundary testing to make sure you are covering everything). Here are some example runs for the program...

```
Input tire pressures: rf lf rr lr
34 36 42 41
Inflation is BAD
Input tire pressures: rf lf rr lr
36 36 42 41
Inflation is GOOD
Input tire pressures: rf lf rr lr
36 40 42 41
Inflation is BAD
```

c. Leap Year. The year 2000 was a leap year. Leap years come every 4th year but most centuries are not leap years. Only every 4th century is a leap year. Write a function that takes a year as a parameter and outputs whether or not it is a leap year. This can be done with a single boolean expression.

d. Write a program that prompts the user to enter an integer and determines whether it is divisible by 5 and 6, whether it is divisible by 5 or 6, and whether it is divisible by 5 or 6, but not both.

e. Write a function that takes as parameters the center *x*-, *y*-coordinates, width, and height of two rectangles and determines whether the second rectangle is inside the first or overlaps with the first, as shown the figure below (the left rectangles are inside, the right rectangles overlap). Test your function to cover all cases and record the tests you performed. Write code to allow a user to enter all the values and then call the function and pop up a pygame window to draw the rectangles the user entered and visually check your results.



2. Chained Conditionals

a. Write a function that accepts a string representation for today's day of the week and an integer representing a number of days after today. The function should display the future day of the week. Here is a sample run. (Hint: Convert the day of the week entered to an integer where 0=Sunday, 1=Monday, etc. then use the modulus operator to figure out the future day)

```
Enter today's day: Monday
Enter the number of days elapsed since today: 3
Today is Monday and the future day is Thursday
Enter today's day: Sunday
Enter the number of days elapsed since today: 31
Today is Sunday and the future day is Wednesday
```

b. Write a function that has parameters for month and year and then displays the number of days in the month. For example, if the user entered month 2 and year 2012, the program should display that February 2012 had 29 days. If the user entered month 3 and year 2015, the program should display that March 2015 had 31 days. You can re-use the Boolean expression you wrote for the leap year question above.

c. (Chained conditionals) Write a function that lets the user guess whether the flip of a coin results in heads or tails. The program randomly generates an integer 0 or 1, which represents head or tail. The function prompts the user to enter a guess (they should enter the words "heads" or "tails") and reports whether the guess is correct or incorrect.

d. (Chained Conditionals) Write a program that asks the user what game they want: heads or tails, rock paper scissors, or guess the dice. Convert the `process_menu` example code to do this, replacing calls to `do_something`, etc. with calls to functions that will play the required games. (For now, these functions can be "stubs", meaning they are just placeholder functions to be filled in later. They could contain a single print statement to let you know that the right function got called.) Then fill in the functions to play the games. Heads or tails is from question 2c, rock paper scissors is from question 3c, and Guess the Dice is a simple game in which the computer rollsl two 6 sided dice, adds them up, then asks the user to guess and gives feedback.

e. A shipping company uses the following function to calculate the cost (in dollars) of shipping based on the weight of the package (in pounds).

$$c(w) = \begin{cases} 3.5, if 0 < w \leq 1 \\ 5.5, if\ 1 < w \leq 3 \\ 8.5, if\ 3 < w \leq 10 \\ 10.5, if\ 10 < w \leq 20 \end{cases}$$

Write a function to calculate the cost from the weight of the package and display a message indicating the shipping cost. If the weight is greater than 50, display the message "the package cannot be shipped."

f. Write a function that simulates picking a card from a deck of 52 cards. Your program should display the rank (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King) and suit (Clubs, Diamonds, Hearts, Spades) of the card. It should print the result (e.g. "Jack of Diamonds" or "3 of Hearts").

3. Nested Conditionals

a. Design, implement and test a function that takes the numeric scores of three darts players (Alice, Harman, and Abdul). Then use nested if statements to determine who had the highest score, and print the name of the winner to the screen. Start by writing a flow chart for the function, then write the code for the function. Finally, write a full set of test cases for the

function (that is, state how many times you would call the function to test it, what arguments you would use for each test, and what you would expect the result to be.

b. Take the code for the `classify_grade` function on Slate. Fix the `classify_grade` function so that if the grade is out of range (less than 0 or greater than 100) it prints an error message of some kind. Otherwise, it should print the grade.

c. Write a function that plays a game of rock-paper-scissors with the user. (Scissors beats paper, paper beats rock, rock beats scissors.) The program randomly generates a number 0, 1, or 2 representing scissor, rock, and paper. The program prompts the user to enter their move (Rock, Scissors or Paper) and displays a message indicating whether the user wins, loses, or draws.

G. Exception Handling

1. Add exception handling to the "Groovy Numbers" exercise from the last handout.
2. Add exception handling to the "Grade Classification" exercise from the last handout. In this case, if the user enters a bad number, move on with the default value of -1.
5. Add exception handling to the "Quadratic Equations" exercise from the handout from two days ago. If the user enters bad numbers, default them to 0. But you should also handle division by zero errors when you are calculating the results.

H. Functions with Return Values

Note that all the functions below should return values instead of printing them. Exercises 3 to 9 adapted from Y. Daniel Liang, *Introduction to Java Programming, 10th Edition*.

1. Do exercise 6.1 in section 6.1 of *Think Python* (write a `compare` function)

2. Do exercise 6.3 in section 6.4 of *Think Python* (is_between function).

3. Do exercise 6.4 in section 6.11 of *Think Python* (tracing function calls).

4. Use the incremental development technique to develop a pair of functions that convert from Celsius to Fahrenheit and from Fahrenheit to Celsius and return the result. The formulas are $F = \frac{9}{5}C + 32$ and $C = \frac{5}{9}(F - 32)$.

   Test the functions like this:

   ```
   print(celcius_to_fahrenheit(Fahrenheit_to_celcius(323)))
   ```

5. Write a program that uses a `for` loop to convert and display the Celsius equivalent of the Fahrenheit temperatures 0 to 100 and vice versa, using the functions you wrote in part 2.

6. Use incremental development to write a function to compute and return the sales commission on a given amount using the formula in the table below. It should accept the sales amount in dollars as a parameter and return the commission amount in dollars.

Sales Amount	Commission Rate
$0.01 - $5,000	8 %
$5,000.01 - $10,000	10 %
$10,000.01 and above	12 %

Note that this is a graduated rate. If the sales amount is $25,000, the commission is $5,000 * 8% + 5,000 * 10% + 15,000*12% = $2,700.

Hint: You'll need nested or chained `if` statements for this one. Start with a commission of 0 and then add to it bit by bit as you go through the `if` statements.

7. Write a Boolean `is_leap_year` function to determine if a given year is a leap. Then write a function called `days_in_year` that returns the number of days in a given year by calling `is_leap_year`.

8. Write a function called `is_valid_triangle` that takes 3 side lengths as a parameter and determines whether they can form a triangle. To be valid, there can be no side that is longer than the other two sides combined. (You did something similar to this before, but the sides were sorted and you printed the result instead of returning it.)

9. Use incremental development to write a function called `triangle_area` that computes and returns the area of a triangle given the three side lengths. If the triangle is not valid, it should return -1 (call the function from question 6 to determine triangle validity).

   You can use Heron's formula to compute the area:

   $A = \sqrt{p(p-a)(p-b)(p-c)}$, where *a*, *b*, and *c* are the sides and $p = \left(\frac{a+b+c}{2}\right)$

10. Use incremental development to write an integer formatting function that takes parameters `n` and `length` (both integers) and returns a string representation of `n`, padded with 0's so that the length is at least `length`. For example, if you call `format_int(34, 5)` it should return the string `'00034'`. If the integer is already more than `length` digits, just return a string containing the integer. Then write a program with a `for` loop to print all the integers from 1 to 10000, padded to a length of 5. (HINT: You may find the built-in functions `len` and `str` very useful.)

11. Write a function called `test_prob` that takes a parameter `p` which is a `float` between 0 and 1 representing a probability. The function should use the `random.randint` function to return `True` with a the given probability (i.e. if p is 0.3 the function should return `True` 30% of the time).

12. Write a function called `test_random` that takes two parameters `p` and `n`, then uses a `for` loop to call `test_prob(p)` `n` times. It should count the number of times `test_prob` returns

`True`, and then compute and return the experimental probability (number of times True came up divided by `n`). If `random.randint` works well, this number should be very close to `p`.

I. Loops

1. Write a function in no more than 5 lines that prints the integers from 1 to 1000. Can you make it count by 10s? Can you make it only display numbers divisible by 5 and 6?
2. Write a function that takes a parameter n and then uses a while loop to get n numbers from the user, adds them up, and returns the total after all n numbers are entered.
3. Try exercises from Think Python (exercise 7.1 from section 7.3 and exercise 7.2 from section 7.5)
5. Write a function that repeatedly asks the user for their name, and then says hello to them using the name they typed. It should keep going until the user enters "quit", and then it should say goodbye and exit.
6. Write a function that chooses a random number from 1 to 10, then asks the user to guess it. It should not let them enter any number out of the range 1 to 10. If they do, it should loop back and ask again. Once they enter a legal number, it should tell them whether they got it right or wrong.

   Challenge: Can you expand the program further so that if the user enters a non number, it gives them an error and loops back?

   Further Challenge: can you expand this program so that when they enter a legal but wrong guess, it tells them "higher" or "lower" and then loops back? (This actually requires a loop inside another loop - a nested loop)

J. "Best So Far" and "Sudden Death" Loops

a. Write a program that asks the user to enter 10 numbers and then outputs the largest number they entered.

b. From *Introduction to Java Programming, 10th Edition*, try exercise 5.14: Write a function that gets two positive integers as parameters and returns the greatest common factor of those two integers. (The greatest common factor is the largest integer that divides both numbers without leaving a remainder. http://en.wikipedia.org/wiki/Greatest_common_divisor.)

c. Can you use your Greatest Common Factor function from part d to reduce a fraction entered by the user? (i.e. 7/21 = 1/3)

d. Write a program that asks the user to enter a password that is at least 8 characters long. Keep looping until they enter a password that meets these criteria. (Use the `len` function to check string length).

e. Write a function called isprime that accepts a positive integer from the user and returns True or False depending on whether the integer is prime. (An integer is prime if and only if it has no factors other than 1 and itself. By definition, the number 1 is *not* a prime number. http://en.wikipedia.org/wiki/Prime_number.)

This is a Sudden Death and Counting problem because it is a search for factors. A number `f` is a factor of `n` if it divides `n` evenly, or in other words if `n % f == 0`. So the basic idea is to search through all the possible factors of n, starting at 2 and going up to n-1. The minute you find a factor, you exit the loop and say "not prime". If you make it all the way up to the end of the possible factors, you should exit the loop and say "prime".

f. Write a program that asks the user to enter a prime number. Then it checks to see if the number is prime (using the loop you wrote for part a) and if it's not prime, asks the user to enter it again and again until they succeed.

K. Nested Loops and Rectangular Structures

1. Write a program that follows the input checking pattern given to ask the user to enter 10 grades between 0 and 100, then display the average grade. Make them repeat any input that is outside the range or not a number. HINT: Write a function to get and return an integer from the user, then expand it so that it catches exceptions and loops back if they enter a bad number, then write a new function that calls the previous one but loops if it returns a number outside of a given range, then use that function in your loop to get the 10 numbers from 0 to 100.

2. Exercise 5.20 from *Introduction to Java Programming, 10th Edition.*
   a. Write code to print out all prime numbers up to 1000. You can re-use the prime number checker you already wrote for this. If working in pairs, look at both your prime number checkers and decide which one would be the best to use. If neither has a working program, use isPrime.java from today's example code file.
   b. Modify the above program to "pretty print" the prime numbers, like this:

```
  2   3   5   7  11  13  17  19  23  29
 31  37  41  43  47  53  59  61  67  71
 73  79  83  89  97 101 103 107 109 113
...
```

   c. Modify the above program so that you get the first 1000 primes, rather than all the primes less than 1000.

3. Exercise 5.33 from *Introduction to Java Programming, 10th Edition.* (Find all perfect numbers.)

4. Try the Think Python exercise 7.3 (test square root algorithm) and 7.5 (sum of series with factorials).

5. Pattern questions from *Introduction to Java Programming, 10th Edition.*
   a. Exercise 5.18, pattern 1: This is rectangular because the row you are in determines how many columns you have to fill in.
   b. Exercise 5.18, patterns 2-4 (if you get time): These are trickier
   c. Harder Exercises: 5.17, 5.19, 5.29

L. Lists

1. Write a function called `createList(value, length)` that returns a list with a given initial `value` and `length`.
    - What happens when the user passes a list to the value parameter?
    - (Tricky) Can you create a new version of `createList` that will work like this instead?

```
>>>createList([1,2,3], 10)
[1,2,3,1,2,3,1,2,3,1]
```

2. Write a function that takes a list parameter and returns one item selected at random.
    - What happens if the user passes a non-list as a parameter? What if the list is empty?
    - Can you fix the function so that it handles these cases?
    - Can you create a new version of the function that returns a random non-empty slice from the list?
3. Write a function that takes a list and an index as a parameter and returns a list that is a copy of the original but with the value of the element at the given index multiplied by 2.
    - Brainstorm a list of all the possible run time errors that could be caused by a programmer passing bad parameters to your function.
    - Make sure you handle all possible error cases.
4. Write a function that takes a list as a parameter and returns a copy of the list with a random non-empty slice removed. Make it handle non-lists and empty lists gracefully.
5. Write a function that accepts four parameters `a`, `s`, `e` and `n`. Create a new list using the slice `a[s:e]` repeated `n` times. Make the function robust against bad parameters. Test thoroughly.
6. Write a function called `fixLength` that takes a list as a parameter and an integer `n`. Return a copy of the list that is exactly `n` elements long. Do this by either cutting of the tail of the list or padding the tail with the value 0.
    - Add a third parameter. If the user passes the value `True`, pad the end of the list, otherwise pad the front.
    - Make the function robust against bad parameter values and test it thoroughly.
7. Write a function called `insertCopy` that accepts two lists (`a` and `b`) and a parameter `i`. Return a copy of list `a` with list `b` inserted just before element `i`.
    - Make the function robust against bad parameter values and test it thoroughly.
8. Write a function called `replaceCopy` that accepts two lists (`a` and `b`) and a parameter `i`. Return a copy of list `a` with elements starting at location `i` replaced with elements from list `b`. The list returned should be the same length as the original list `a`.

```
>>>replaceCopy([1, 2, 3, 4], [0, 0], 1)
[1, 0, 0, 4].

>>>replaceCopy([1, 2, 3, 4], [0, 0, 0, 0], 1)
[1, 0, 0, 0].
```

9. (Tricky) Write a function that creates a list containing a three word secret message hidden in 20 elements. Choose the locations of the three words randomly. Then in a loop, ask the user to

enter 3 index values and give them hints each time telling them how many they found and exit when they have found all three, revealing the message.

- (Trickier) Modify the function so the user has to find all 3 words in the correct order. When you give hints, tell them how many they found in the correct position and how many they found in the incorrect position.

M. List Processing

Quick List Processing Exercises

1. Write a function that prints the elements of a list in forwards order

2. Write a function that prints the elements of a list in reverse order.

2. Write a function that prints every n^{th} element of a list, where n is a parameter.

3. Write function that returns a list of 1000 items filled with the numbers from 1 to 1000.

Less Quick List Processing Exercises

* = adapted from Y. Daniel Liang, *Introduction to Java Programming, 10th Edition*.

0. Classify each of the functions you are asked to write below as Map, Filter or Reduce functions.

1. A function that returns the average of a list of numbers

2. A function that takes a list of integers and returns a new list of the same length containing just the last digit of each integer.

3. A function that takes a list of words and a minlength parameter, then returns a new list with all the short words (less than minlength) removed.

4. A function that returns the highest number in a list of numbers. (Uses the BestSoFar pattern.)

5. A function that searches through a list for a given key and returns the index of the key if found or -1 if not found. (Uses the SuddenDeath pattern.)

6. * A function that takes a list of numbers as a parameter and then returns an integer that represents how many of the numbers in the list are greater than or equal to the average.

7. A function that takes a list of numbers as a parameter, computes the average and returns a new list with all the numbers less than average removed.

8. * A function that finds the best score from a list of student grades (you can call the function from exercise 1) and then returns a list representing the letter grade for each student. It should be an A if a student's score is greater than or equal to the best score minus 10. B for >= best-20, C for >= best-30, D for >= best-40, F otherwise.

```
>>>letterGrades([40, 55, 70, 58])
['C', 'B', 'A', 'B']
```

9. A function that takes a list and returns a new version of the list with all duplicate elements removed.

10. (From *Think Python*) A function that takes a list of numbers and returns the cumulative sum; that is, a new list where the i^{th} element is the sum of the first $i + 1$ elements from the original list. For example, the cumulative sum of [1, 2, 3] is [1, 3, 6].

11. A function that takes a list parameter and returns a copy of the list in reverse order.

12. (From *Think Python*) A function `issorted` that returns `True` if its list parameter is sorted in non-decreasing order and `False` otherwise.

Even Less Quick List Processing Exercises

1. Write a function that plays a guessing game with the user. It should have a list with at least 20 different items, and then should start randomly guessing from the list (make sure it's random) until it gets it right. The output should look something like this:

```
Hi, let's play a guessing game. Think of an animal and
press a key when you are ready.
Is it a lion?
no
Is it an ostrich?
no
Is it a poodle?
yes
I win!!!
```

2. Modify your solution to the last question so the program never guesses the same thing twice and gives up after it's tried everything it knows.

3. *Write a function that reads integers between 1 and 100 entered by the user. When the user enters a 0, stop and return the number of times each number was entered:

```
Enter ints from 1 to 100: 2 5 6 5 4 3 23 43 2 0
2 was entered 2 times
3 was entered 1 time
4 was entered 1 time
5 was entered 2 times
6 was entered 1 time
23 was entered 1 time
43 was entered 1 time
```

4. Write a function that takes a parameter n, then simulates the rolling of a 20-sided die n times and prints a histogram of the outcomes (a histogram tells you how many times each outcome came up)

5. Write a function similar to the one in the previous question, but simulate n rolls of two 6-sided dice (i.e. the outcomes are 2 through 12)

6. *Write a function that takes a list of numbers and returns the standard deviation of the list of numbers. You get the standard deviation by computing the mean (i.e. average) and then summing x_i – mean for each item x_i in the list, dividing the total by n-1 and then taking the square root.

$$standard\ deviation = \sqrt{\frac{\sum_{i=1}^{n} x_i - mean}{n-1}}$$

7. *Write a function that returns a random integer between 1 and 52, but skips over the numbers in the list given as a parameter. (This could be used to simulate choosing a card from a deck of cards where some cards have already been drawn.)

8. Write a function that implements a game of war between two players using the above function. On each round, each player draws a card and the high card wins. Each card drawn is added to a list that is used to make sure it won't be drawn again. Keep going until the deck is empty. (Hint: if the card number is c, use c // 13 for the suit number – e.g. 0=hearts, 1=clubs, etc. – and c%13 for the card's rank – e.g. 0 = ace, 1 = 2, 2 = 3, …, 10 = Jack, 11 = Queen, 12 = King)

9. *The most efficient way to check if a number is prime is to check the prime number divisors <= the square root. So to check if 100 is prime, just check the primes 2, 3, 5 and 7. If you find one that divides in evenly, it's not prime. Write a function that displays the first n prime numbers using this approach to determining if a number is prime. You can use a list to store all the prime numbers you have found so far.

Optional String Processing Exercises (see section 10.9)

* = adapted from Y. Daniel Liang, *Introduction to Java Programming, 10th Edition*.

1. (From *Think Python*) Two words are anagrams if you can rearrange the letters from one to spell the other. Write a function called `is_anagram` that takes two strings and returns True if they are anagrams. (Hint: There's a solution to this that doesn't require you to write a loop.)

2. Write a function that takes a string parameter containing a sentence and returns a float representing the average length of words in the string (you can assume every character except the space character is part of a word).

3. A function that takes a string parameter containing a sentence and returns a string with the original words of the sentence in reverse order (e.g. "hi there" → "there hi").

4. A function that takes a string parameter containing a sentence and returns a string with the original words reversed (e.g. “hi there” → “ih ereht”).

5. A function that returns a “telegraphic” form of a sentence by removing all words shorter than 4 characters. (e.g. “I am entering the center of the storm” → “entering center storm”)

6. *Implement a game of hangman, choosing the word randomly from a predefined a list of possible words.

```
Enter a letter in word *******: p
Enter a letter in word p******: r
Enter a letter in word pr**r**: p
      You already guessed p
Enter a letter in word pr**r**: n
      The word does not have an n
Enter a letter in word pr**r**: m
Enter a letter in word pr**r*m: a
Enter a letter in word pr**ram: o
Enter a letter in word pro*ram: g
You got it! The word is program. You missed 1 time.
Go again (y/n)? n
```