

FYS 102: Bioinformatics  
Homework 4: Python Practice  
Due Friday, October 5  
(Start now!)

---

1. Write a Python function `PrintNumbers` that takes two integer parameters and then uses a for loop to print all the integers in that range. For example, calling `PrintNumbers(6, 13)` should print

```
6 7 8 9 10 11 12 13
```

Hint: To print something without jumping to the next line, append the something with a comma. For example,

```
print x,
```

2. Write a Python function `Search` that takes a list and a list item as parameters and then returns 1 if the item is in the list, or 0 otherwise. Your function must use a for loop (*not* a built-in list function). For example, if the variable `list` is `[4, 3, 8, 2, 0]`, then `Search(list, 8)` should return 1 while `Search(list, 1)` should return 0.
3. Write a Python function `Count10` that takes a list of numbers as a parameter and returns the number of 10's in the list. For example, `Count10([4, 13, 10, 10, 3, 10, 13])` should return 3.
4. An infinite series is an infinite sum of terms that typically follows some pattern. Infinite series are sometimes used to approximate important values. The following is an infinite series that calculates an approximation of  $\pi$ :

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots$$

Write a Python function `ApproxPi` that computes the value of this sum with any given number of terms. (The number of terms should be given as a parameter.) This is really just a running sum like those we have seen before, but you will need to think about how to change the denominator and alternate the sign of the term each time through the loop. Use decimal points in your values to ensure that real division is used (instead of integer division). For reference, here is what your function should get for increasing numbers of terms:

```
ApproxPi(5):      3.33968253968
ApproxPi(10):     3.04183961893
ApproxPi(100):    3.13159290356
ApproxPi(1000):   3.14059265384
ApproxPi(10000):  3.14149265359
```

5. Write a Python function `Rectangle` that prints a rectangle of given width and height with a given character. Your function should take the width, height, and character as parameters. For example, `Rectangle(4, 5, '*')` should print

```
* * * * *
* * * * *
* * * * *
* * * * *
```

(The spaces between the characters are fine; using the comma in a print statement as in problem 1 forces this to happen.) You will need to use two nested loops for this problem: a loop counting columns inside a loop counting rows.

## 6. Extra credit!

- (a) Write a Python function `FindSmallest` that takes a list of numbers as a parameter and returns the index of the smallest element. For example, if `list = [4, 3, 5, 1, 0]`, then `FindSmallest(list)` should return 4, the index of element 0.
- (b) Write a Python function `Swap` that swaps two elements in a list. The function should take 3 parameters: a list and the indices of the two values to swap. For example, if `list` is defined as in (a), then `Swap(list, 1, 4)` should return the list `[4, 0, 5, 1, 3]`. Hint: you will need an extra variable in your function to temporarily hold one of the elements.
- (c) A *selection sort* is an algorithm that sorts a list by repeatedly finding the smallest element in the unsorted part of the list and swapping into its correct position. For example, suppose we wanted to sort the list given above: `[4, 3, 5, 1, 0]`. First, the algorithm finds the index of the smallest element in the list (see (a)) and swaps it (see (b)) with the first element, giving

`[0, 3, 5, 1, 4]`

Then it repeats this process with the unsorted portion of the list (indices 1 through 4 in this case or `list[1:]`). This will give

`[0, 1, 5, 3, 4]`

Repeating again with `list[2:]` gives

`[0, 1, 3, 5, 4]`

And finally, performing this process on `list[3:]` gives

`[0, 1, 3, 4, 5]`

Write a Python function `SelectionSort` that uses the selection sort algorithm to sort a list. Your function should take the list as a parameter. Your function will consist of a for loop containing two appropriate function calls: one to `FindSmallest` and one to `Swap`. Of course, you are not permitted to use any built-in list functions here (like `sort`).

Start early and have fun!