

Due by Tuesday 9/2 at 9:00

1. Get the html or pdf file of the Python tutorial from

<http://docs.python.org/download.html>.

Read sections 1, 2, 3, 4.1, 4.2, 4.3, 4.6, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 or as needed to answer the questions 4 and 6 that require basic understanding of Python. Work through examples in the tutorial. (Nothing to be handed in for this question.)

2. Consider the following algorithm:

```
def isprime(n):
    for i = 2, ..., sqrt(n):
        if n is divisible by i return false
    return true
```

What does the algorithm do? What is the running time of the algorithm expressed as a function of  $n$ ? Suppose  $n$  is represented as a binary number. Is this algorithm polynomial? Is it exponential?

3. Consider the function `improve` defined as follows:

```
def improve(f):
    known = {}
    def compute(arg):
        if arg not in known
            known[arg] = f(arg)
        return known[arg]
    return compute
```

Take the first function we wrote for computing Fibonacci numbers,

```
def fib(n):
    if n==0 or n==1:
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

enter or load the definitions of `fib` and `improve` in your Python interpreter and then try the following two experiments:

- (a) call `fib(30)` and then `fib(40)`
- (b) when `fib(40)` halts or you interrupt it, type `fib = improve(fib)` and then call `fib(100)`.

What is going on? Explain what improve does and how it works.

*[Useful Python background: definition of function, functions as return values, dictionaries.]*

4. Let  $T(n)$  be defined recursively as

$$T(n) = \begin{cases} 2, & \text{if } n = 2 \\ 2T(n/2) + n, & \text{if } n = 2^k, k > 1 \end{cases}$$

Show by induction that  $T(n) = n \lg n$  for all  $n$  that are powers of 2.

5. Describe an  $\Theta(n \lg n)$ -time algorithm that, given a set  $S$  of  $n$  real numbers and another real number  $x$ , determines whether or not there exist two elements in  $S$  whose sum is exactly  $x$ .