Terms to know

Computable: a problem is computable if it is possible for it to be solved with a computer program

Halt: a function halts on an input if it eventually returns (or crashes, so long as it does not loop infinitely) when called on that input

Hang: a function hangs on an input if it loops infinitely when called on that input

Proof

We want to show that it is impossible to tell whether or not an *arbitrary* function halts on a given input. We will prove this by contradiction.

Assume that we have a function, halts(f, i), that takes two arguments: f, which is a function that takes one argument, and i, which is some argument. halts returns 1 if f(i) halts, and 0 if f(i) hangs. Halts does this without actually calling f(i), because if we call f(i) then there is no way to distinguish between the function taking a very long time or just hanging. (Right now, we are limiting ourselves to functions that take just one argument; it is trivial to expand the argument to functions that take different numbers of arguments)

Now, we number all possible functions of one argument, as f1, f2, f3, ...
We also number all possible inputs, as 1, 2, 3, ...

We construct a table:

	1	**2**	**3**	**...**
f1	1	1	0	
f2	0	1	0	
f3	0	0	0	
...				

Where the values in the table are the output of halts, called on the function in that row and the input in that column.

Now, we write another function:

```
def nefariousFunction(f):
    if halts(f, f):
        while(True): pass # hang
    else:
        return # halt
```

Since this function is written in valid Python, and takes one argument, it must be a row in our table. It must also be a column, since we can have functions as arguments. Let's say it appears at row *k* and column *j*. What value appears in our table at row *k* and column *j*?

We see that no value can appear at this cell in the table:

> If a 1 appeared there, that would mean that nefariousFunction(nefariousFunction) would halt. But, if nefariousFunction(nefariousFunction) halts, then it hangs, so a 0 would need to be in the table.

> If a 0 appeared there, that would mean that nefariousFunction(nefariousFunction) does not halt. But, if nefariousFunction(nefariousFunction) does not halt, then it returns, so a 1 would need to be in the table.

This breaks our table: we said our table counted all of the functions of one argument, but the nefariousFunction is a function of one argument that is not included in our table.

This means that one of our assumptions was wrong. Since all of the steps we took in our proof were legal, the only assumption that can be wrong is the assumption that halts(f, i) exists. Therefore, halts(f, i) does not exist.

This proof shows that we cannot determine if an arbitrary function halts.