1.12. Defining Functions

The earlier example of procedural abstraction called upon a Python function called `sqrt` from the math module to compute the square root. In general, we can hide the details of any computation by defining a function. A function definition requires a name, a group of parameters, and a body. It may also explicitly return a value. For example, the simple function defined below returns the square of the value you pass into it.

```
>>> def square(n):
...     return n**2
...
>>> square(3)
9
>>> square(square(3))
81
>>>
```

The syntax for this function definition includes the name, `square` , and a parenthesized list of formal parameters. For this function, `n` is the only formal parameter, which suggests that `square` needs only one piece of data to do its work. The details, hidden “inside the box,” simply compute the result of `n**2` and return it. We can invoke or call the `square` function by asking the Python environment to evaluate it, passing an actual parameter value, in this case, `3` . Note that the call to `square` returns an integer that can in turn be passed to another invocation.

We could implement our own square root function by using a well-known technique called “Newton’s Method.” Newton’s Method for approximating square roots performs an iterative computation that converges on the correct value. The equation $newguess = \frac{1}{2} * (oldguess + \frac{n}{oldguess})$ takes a value n and repeatedly guesses the square root by making each $newguess$ the $oldguess$ in the subsequent iteration. The initial guess used here is $\frac{n}{2}$. Listing 1 shows a function definition that accepts a value n and returns the square root of n after making 20 guesses. Again, the details of Newton’s Method are hidden inside the function definition and the user does not have to know anything about the implementation to use the function for its intended purpose. Listing 1 also shows the use of the # character as a comment marker. Any characters that follow the # on a line are ignored.

Listing 1

```
def squareroot(n):
    root = n/2    #initial guess will be 1/2 of n
    for k in range(20):
        root = (1/2)*(root + (n / root))

    return root
```

```
>>>squareroot(9)
3.0
>>>squareroot(4563)
67.549981495186216
>>>
```

(ExceptionHandling.html) (ObjectOrientedPr

Self Check

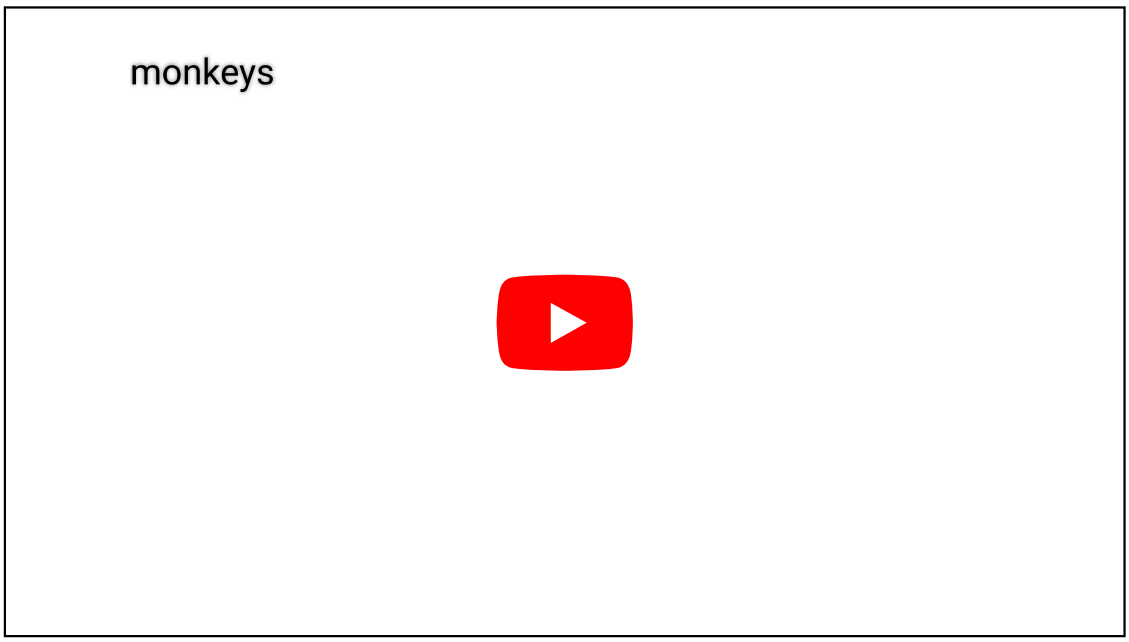
Here's a self check that really covers everything so far. You may have heard of the infinite monkey theorem? The theorem states that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type a given text, such as the complete works of William Shakespeare. Well, suppose we replace a monkey with a Python function. How long do you think it would take for a Python function to generate just one sentence of Shakespeare? The sentence we'll shoot for is: "methinks it is like a weasel"

You're not going to want to run this one in the browser, so fire up your favorite Python IDE. The way we'll simulate this is to write a function that generates a string that is 28 characters long by choosing random letters from the 26 letters in the alphabet plus the space. We'll write another function that will score each generated string by comparing the randomly generated string to the goal.

A third function will repeatedly call generate and score, then if 100% of the letters are correct we are done. If the letters are not correct then we will generate a whole new string.To make it easier to follow your program's progress this third function should print out the best string generated so far and its score every 1000 tries.

Self Check Challenge

See if you can improve upon the program in the self check by keeping letters that are correct and only modifying one character in the best string so far. This is a type of algorithm in the class of 'hill climbing' algorithms, that is we only keep the result if it is better than the previous one.



Activity: 1.12.1 YouTube (monkeyvid)

You have attempted 1 of 2 activities on this page

(ExceptionHandling.html) (ObjectOrientedPr

user not logged in