Editor

User Input via the Command Line

Section 22.8

Completed

‹ back to module

Skip straight to editor/terminal

If you've ever used a security tool that's been built in Python, you may already be familiar with the idea that you can pass user input to a Python script at the point you choose to run it from the command line. If you're building a program that you mostly expect people to run from a terminal, it may make sense to capture your user input this way.

In order to do this, we'll first have to import a separate library.

Importing a library

Python comes with a standard library of additional components we can use in our code. They have been split out into smaller groups of functionality that we have to import in order to make our programs more efficient. If all the possible bits of functionality were always included, our programs would take longer to run. So the 'default' set of Python functionality includes only the core tools we need basically all the time, while other useful pieces of functionality are split up into useful libraries or modules that we can import and use at any time.

You can find a full list of the tools available in Python 2's standard library in the official Python 2 documentation here.

In order to allow user input to be passed via the command line at the program's run time, we need one of the Python Runtime Services libraries called "System-specific parameters and functions". This is the `sys` library, and to import and use it in our program we'll have to start our program off with `import sys`. You should always import the libraries or modules you intend to use in your program at the start. It's good practice to declare your dependencies upfront.

How command line arguments are passed to the program

Let's start by just passing in some user input and seeing what kind of format we get out the other side. When using a new function you've never used before, a good way to start understanding the ins and outs of how it works is by outputting what you get back to the screen. In order to access the arguments we pass in when we run our program via the command line, we use `sys.argv`.

```
import sys

arguments = sys.argv
print(arguments)
```

Let's start by running our program with some test arguments and see what we get as output when we print them. Then, we know what kind of data structure we're dealing with when we use `sys.argv`.

```
$ python program.py test1 test2 test3
['program.py', 'test1', 'test2', 'test3']
```

A few things we notice right away about our output:

1. We've seen this structure before: thos `[]` brackets indicate this is a list. The data we get is in a list form, so we can interact with it as a list.
2. The first item in our list is the name of the python program we ran, in this case, `program.py`
3. The arguments we pass through are added into the list, in the order that we run them.

Ok, that's some solid information gathering. Let's try a few more things and see what happens.

```
$ python program.py 1 false this='that' x=2
['program.py', '1', 'false', 'this=that', 'x=2']
```

We've given this another test run to see if everything we throw at it will be processed as a list, and to see if it will take all different data types and turn them into strings. As we can see in this example, that's exactly what it does: everything we pass in as an argument will become available to us in our program as a list of strings, with our program filename being the very first string in the list.

Using command line arguments in our program

Agent M sometimes needs to pick code names for certain secret projects the team is working on. After many years, she has started to run out of ideas so she needs a program to help her select code names for new projects. We'll create a list of words suitable to be code names, and using a command line argument we'll specify the number of words we want to use from the list to generate our random code name:

```
import random

# Pick a random word from a provided list
def pick_random_word(list):
    return random.choice(list)

# Get a code name made up of the number of words specified
def get_code_name(list, num_words):
    code_name = ''

    for x in range(1,num_words+1):
        word = pick_random_word(list)
        code_name += word + ' '

    return code_name.rstrip()

# List of words to use
word_list = ['Aurora', 'Avalanche', 'Blizzard', 'Cyclone', 'Eagle', 'Edison', 'Frost',
'Hawk', 'Hexagon', 'Hornet', 'Medusa', 'Neptune', 'Orion', 'Osprey', 'Plato', 'Portal',
'Raven', 'Sand', 'Shadow', 'Storm', 'Sunset', 'Thunder', 'Vector', 'Vista', 'Vortex',
'Volcano']

# Create a code name and print it to the screen
code_name = get_code_name(word_list, 2)
print(code_name)
```

This simple little program allows us to generate code names using words picked at random from a list we provide, and provides us a way to specify how many words we want to use for our code name.

Now, let's add in the ability to specify the number of words from the command line, where we run the script. Don't forget the lesson we learned from the section on user input prompts, where we need to make sure we check the input we receive is usable by our program: in this case, we need an integer. We've already included a default parameter in our `getCodeName()` definition, so if we don't get an input we can use, we'll use our default number of words, which is 1.

```
import random
import sys

# Pick a random word from a provided list
def pick_random_word(list):
    return random.choice(list)

# Get a code name made up of the number of words specified
def get_code_name(list, num_words):
    if num_words.isdigit() == False:
        return 'Error: incorrect argument provided. You must provide an integer.'

    num_words = int(num_words)
    code_name = ''

    for x in range(1,num_words+1):
        word = pick_random_word(list)
        code_name += word + ' '

    return code_name.rstrip()

# List of words to use
word_list = ['Aurora', 'Avalanche', 'Blizzard', 'Cyclone', 'Eagle', 'Edison', 'Frost',
'Hawk', 'Hexagon', 'Hornet', 'Medusa', 'Neptune', 'Orion', 'Osprey', 'Plato', 'Portal',
'Raven', 'Sand', 'Shadow', 'Storm', 'Sunset', 'Thunder', 'Vector', 'Vista', 'Vortex',
'Volcano']

# Retrieve the command line argument
words_to_pick = sys.argv[1]

# Create a code name and print it to the screen
code_name = get_code_name(word_list, words_to_pick)
print(code_name)
```

We've only added a couple of things to our code here to get it working with command line arguments.

In the `get_code_name()` function definition, we added a check to make sure our user input is something we can turn into an integer using our old friend the `.isdigit()` method again. If it fails this check, we immediately return with an error that gets printed to the screen, and the rest of this function never runs. (Remember: when we return in a function it exits the function immediately.) If this conditional passes, the first thing we do next is change this string into an integer so we can use it in our loop as we did before.

The only other line we've added is just below where we create our list of words, which grabs the command line argument so we can pass it through to the `get_code_name()` function.

Now we can run our program via the command line and pass in our user data, which is the number of random words we want our code name to include. Here is an example from running the program three times, once requesting a code name with 2 words, once with 3 words, and once using a string which causes our error to appear.

```
$ python program.py 2
Sunset Volcano
$ python program.py 3
Shadow Blizzard Orion
$ python program.py test
Error: incorrect argument provided. You must provide an integer.
```

One last thing we need to take into account: what happens if the person running our program doesn't know they have to add an argument?

```
$ python program.py
Traceback (most recent call last):
  File "program.py", line 29, in <module>
    words_to_pick = sys.argv[1]
IndexError: list index out of range
```

Hmmm... that's no good. Let's adjust our code one more time to provide a helpful error for this case as well.

```
import random
import sys

# Pick a random word from a provided list
def pick_random_word(list):
    return random.choice(list)

# Get a code name made up of the number of words specified
def get_code_name(list, num_words):
    if num_words.isdigit() == False:
        return 'Error: incorrect argument provided. You must provide an integer.'

    num_words = int(num_words)
    code_name = ''

    for x in range(1,num_words+1):
        word = pick_random_word(list)
        code_name += word + ' '

    return code_name.rstrip()

# List of words to use
word_list = ['Aurora', 'Avalanche', 'Blizzard', 'Cyclone', 'Eagle', 'Edison', 'Frost',
'Hawk', 'Hexagon', 'Hornet', 'Medusa', 'Neptune', 'Orion', 'Osprey', 'Plato', 'Portal',
'Raven', 'Sand', 'Shadow', 'Storm', 'Sunset', 'Thunder', 'Vector', 'Vista', 'Vortex',
'Volcano']

if len(sys.argv) > 1:
    # Retrieve the command line argument
    words_to_pick = sys.argv[1]

    # Create a code name and print it to the screen
    code_name = get_code_name(word_list, sys.argv[1])
    print(code_name)
else:
    print('Error: You must provide the number of words as an argument.')
```

And let's test our different cases one more time on the command line to make sure we get what we expect in each situation.

```
$ python program.py 1
Hexagon
$ python program.py 3
Eagle Sand Cyclone
$ python program.py test
Error: incorrect argument provided. You must provide an integer.
$ python program.py
Error: You must provide the number of words as an argument.
```

Nice! Agent M will be very pleased.

Completed

‹ back to module