12.6. Decoding a Function

In general, when you see a function definition you will try figure out what the function does, but, unless you are writing the function, you won't care *how it does it*.

For example, here is a summary of some functions we have seen already.

- `input` takes one parameter, a string. It is displayed to the user. Whatever the user types is returned, as a string.
- `int` takes one parameter. It can be of any type that can be converted into an integer, such as a floating point number or a string whose characters are all digits.

Sometimes, you will be presented with a function definition whose operation is not so neatly summarized as above. Sometimes you will need to look at the code, either the function definition or code that invokes the function, in order to figure out what it does.

To build your understanding of any function, you should aim to answer the following questions:

1. How many parameters does it have?
2. What is the type of values that will be passed when the function is invoked?
3. What is the type of the return value that the function produces when it executes?

If you try to make use of functions, ones you write or that others write, without being able to answer these questions, you will find that your debugging sessions are long and painful.

The first question is always easy to answer. Look at the line with the function definition, look inside the parentheses, and count how many variable names there are.

The second and third questions are not always so easy to answer. In Python, unlike some other programming languages, variables are not declared to have fixed types, and the same holds true for the variable names that appear as formal parameters of functions. You have to figure it out from context.

To figure out the types of values that a function expects to receive as parameters, you can look at the function invocations or you can look at the operations that are performed on the parameters inside the function.

Here are some clues that can help you determine the type of object associated with any variable, including a function parameter. If you see…

- `len(x)`, then x must be a string or a list. (Actually, it can also be a dictionary, in which case it is equivalent to the expression `len(x.keys())`. Later in the course, we will also see some other sequence types that it could be). x can't be a number or a Boolean.
- `x - y`, x and y must be numbers (integer or float)
- `x + y`, x and y must both be numbers, both be strings, or both be lists
- `x[3]`, x must be a string or a list containing at least four items, or x must be a dictionary that includes 3 as a key.
- `x['3']`, x must be a dictionary, with '3' as a key.
- `x[y:z]`, x must be a sequence (string or list), and y and z must be integers
- `x and y`, x and y must be Boolean
- `for x in y`, y must be a sequence (string or list) or a dictionary (in which case it's really the dictionary's keys); x must be a character if y is a string; if y is a list, x could be of any type.

Check your understanding: decode this function definition

UFunc-1-1: How many parameters does function cyu3 take?

```
def cyu3(x, y, z):
   if x - y > 0:
      return y -2
   else:
      z.append(y)
      return x + 3
```

- A. 0
- B. 1
- C. 2
- [✓ D. 3]
- E. Can't tell

Check me Compare me

✓ x, y, and z.

Activity: 1 -- Multiple Choice (question200_1_1)

UFunc-1-2: What are the possible types of variables x and y?

```
def cyu3(x, y, z):
   if x - y > 0:
      return y -2
   else:
      z.append(y)
      return x + 3
```

- [✓ A. integer]
- [✓ B. float]
- C. list
- D. string
- E. Can't tell

Check me Compare me

✓ Correct.
A. x - y, y-2, and x+3 can all be performed on integers.
B. x - y, y-2, and x+3 can all be performed on floats.

Activity: 2 -- Multiple Choice (question200_1_2)

UFunc-1-3: What are the possible types of variable z?

```
def cyu3(x, y, z):
    if x - y > 0:
        return y -2
    else:
        z.append(y)
        return x + 3
```

- [ ] A. integer
- [ ] B. float
- [✓] C. list
- [ ] D. string
- [ ] E. Can't tell

Check me   Compare me

✓ Correct.
C. append can be performed on lists.

Activity: 3 -- Multiple Choice (question200_1_3)

UFunc-1-4: What are the possible types of the return value from cyu3?

```
def cyu3(x, y, z):
    if x - y > 0:
        return y -2
    else:
        z.append(y)
        return x + 3
```

- [✓] A. integer
- [✓] B. float
- [ ] C. list
- [ ] D. string
- [ ] E. Can't tell

Check me   Compare me

✓ Correct.
A. y-2 or x+3 could produce an integer.
B. y-2 or x+3 could produce a float.

Activity: 4 -- Multiple Choice (df_question200_1_3)

You have attempted 5 of 4 activities on this page

12.5. Returning a value from a function">

eturning a value from a function">

12.7. A function that accumulates">

✓ Completed. Well Done!

12.7. A function that accumulates">Next Section - 12.7. A function that accumulates

| Back to top