Introduction_to_Functions

May 1, 2022

In this notebook I outline the basics of creating and using functions in Python. I will brefly describe what a function is, demonstrate a function's basic syntax, and then dive deep into function usage. The objective is for students to grasp the conept of how functions can be used to help organize and reuse their code.

1 Prerequisites

1.1 Import the necessary packages.

First we will load the packages necessary to complete our demonstration. We will primarily be using the Pandas package to help read in our data.

[1]:
```
import pandas as pd # For loading the CSV file into a dataframe.
```

1.2 Load some data.

For this demonstration, we will be using the Titanic dataset, which is readily available on Kaggle.com. This dataset details the manfest of the RMS Titannic. It contains passenger demographic information as well as information relating to the passenger's economic status. For the purposes of this demonstration we will focus on how a function works in relation to the dataset.

To load the data we will read the titannic.csv file into dataframe using the Pandas package. We will cover dataframes later on in the course.

[2]:
```
data = pd.read_csv('titanic.csv')
```

Once we have our dataframe loaded let's inspect it briefly to get a sense of the collumns and rows.

[3]:
```
data.head()
```

[3]:

	PassengerId	Survived	Pclass	\
0	892	0	3	
1	893	1	3	
2	894	0	2	
3	895	0	3	
4	896	1	3	

	Name	Sex	Age	SibSp	Parch	\
0	Kelly, Mr. James	male	34.5	0	0	

```
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

[4]:
```
data.tail()
```

[4]:
```
     PassengerId  Survived  Pclass                          Name     Sex  \
413         1305         0       3              Spector, Mr. Woolf    male
414         1306         1       1    Oliva y Ocana, Dona. Fermina  female
415         1307         0       3    Saether, Mr. Simon Sivertsen    male
416         1308         0       3             Ware, Mr. Frederick    male
417         1309         0       3        Peter, Master. Michael J    male

      Age  SibSp  Parch              Ticket      Fare Cabin Embarked
413   NaN      0      0           A.5. 3236    8.0500   NaN        S
414  39.0      0      0            PC 17758  108.9000  C105        C
415  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416   NaN      0      0              359309    8.0500   NaN        S
417   NaN      1      1                2668   22.3583   NaN        C
```

As we can see we now have our data in a neat, concise format. We are ready to dive in!

2 Function Fundamentals

At the bare minimum function is defined as a collection of related statements that perform a specific task. The code in a function is reusable, meaning it can be called repeatedly to produce the same result over and over again. Functions are a very useful component of the related concept of object oriented programming.

2.1 Basic Function Syntax

A function actually has seven parts, with some optional:

1. The def keyword which marks the start of the function head.
2. A function name to identify the function with.
3. Optional parameters or arguments through which we pass values to a function.
4. A colon (:) to mark the end of the end of the function header.
5. An optional documentation string (commonly referred to as a docstring) to help describe wha
6. One or more valid Python statements that make up the function body.
7. An optional return statement to return a value from the function. Note that there can be mu

Here is what a function looks like.

[5]:
```
def functionName (parameter = 'test'):
    '''
    This is the docstring for this function.
    It serves two purposes.
        1. It describes what the function is doing.
        2. It is what is displayed when the function is called with the
            help function.

    This function capitolizes all characters in your sting.
    '''

    ## Valid Python statements go here. ##
    print('Converting ', parameter, ' to all caps!', sep = '')

    # We save the parameter string to a variable.
    string = parameter

    # Lastly we return the variable value from the function in all caps.
    return string.upper()
```

A couple of things to note here.

Notice how the parameter has a string assigned to it. This is the default value of the parameter. If the function is called without a parameter specified, then the parameter will be assigned with its default value for that instance of the function run.

The docstring is, while optional, used to provide information when calling the help function on your function. So it's generally best practice to include one along with regular comments in the statement section so that your code is readable and managable.

Note how I passed the parameter to a variable in the function and then used that variable to manipulate the data contained therein. This is a standard method of passing data into your functions.

Lastly, I return the capitolized version of the string that was passed into the function at the end. For usage, the function must be assigned to a variable for it to be output.

If we wish, we can also have multiple parameters in the function.

[6]:
```
def functionName2 (parameter = 'test', parameter2 = 'test2'):
    '''
    This is the docstring for this function.
    It serves two purposes.
        1. It describes what the function is doing.
        2. It is what is displayed when the function is called with the
            help function.
```

```
    This function capitolizes all characters in your sting.
    '''

    ## Valid Python statements go here. ##
    print('Converting ', parameter, ' to all caps!', sep = '')

    # We save the parameter string to a variable.
    string = parameter
    string2 = parameter2

    # Lastly we return the variable value from the function in all caps.
    return string.upper(), string2
```

2.2 Function Usage

Let's check out the documentation for the function.

[7]:
```
help(functionName)
```

```
Help on function functionName in module __main__:

functionName(parameter='test')
    This is the docstring for this function.
    It serves two purposes.
        1. It describes what the function is doing.
        2. It is what is displayed when the function is called with the
            help function.

    This function capitolizes all characters in your sting.
```

Notice that the function's docstring was displayed. This can be usedful in learning about a new function that you haven't used before. It's important to keep your functions well documented so other developers can pick up where you left off. This is part of that.

Let's call the function WITHOUT a parameter.

[8]:
```
functionName()
```

```
Converting test to all caps!
```

[8]:
```
'TEST'
```

Two things happened here. First, the default value was used for the parameter, and then passed through the function, outputting it in the console. Notice that I didn't assign it to a variable. Second, the contents of the print statement was printed out. What I am illustrating here is, that print can be very useful in diagnosing bfunction behavior.

Let's try it again, only this time I'm going to assign the value returned by the function to a variable, and then view that variable.

[9]:
```
variable = functionName()
variable
```

```
Converting test to all caps!
```

[9]:
```
'TEST'
```

Now we can call the variable in our code for reuse.

[10]:
```
print('This is a ', variable, '!', sep = '')
```

```
This is a TEST!
```

We can also pass other values to the function.

[11]:
```
variable2 = functionName('Sky')
print('My name in all caps is, ', variable2, '.', sep = '')
```

```
Converting Sky to all caps!
My name in all caps is, SKY.
```

Let's call our second function with two variables.

[12]:
```
variable3, variable4 = functionName2('foo', 'bar')
print(variable3)
print(variable4)
```

```
Converting foo to all caps!
FOO
bar
```

Note how we were able to pass in two strings, and return two values.

The point is, functions are a very convenient means of performing repetative tasks efficiently.

2.3 Scope

Let's now, briefly talk about scope in relation to function variables. In Python, variables can have two different layers of scope: Global and Local.

Variables in the global scope are callable through out the environment. This includes inside functions and other objects. Local scope is different in that variables with local scope are only callable from within the object they are contained in. I will illustrate this for you.

Let's define a global variable.

[13]:
```
x = 300
```

Calling x we can see it's availibiliy in the environment.

[14]:
```
print(x)
```

```
300
```

Now I'm going to define a function and pass in my global variable x. Inside my function, I will also assign a local variable.

[15]:
```
def plusFive(x):
    '''
    This sample function takes the integer in from the x variable, and adds␣
 ↪five to it.
    '''

    # Now let's define a local variable.
    y = 5

    # Add y to x.
    return x + y
```

Now let's call our function with the x varable and see what the result is.

[16]:
```
result = plusFive(x)
print(result)
```

```
305
```

Here we've demonstrated that we can pass our global variable x to the local scope insode our function. What will happen if we now try to call our local variable y from the global environment?

[17]:
```
try:

    print(y)

except:

    print("Local variable y wasn't found in the global scope.")
```

```
Local variable y wasn't found in the global scope.
```

We can see how the variables we define inside of functions can't be called from outside the object.

3 Practical Function Demonstration.

We will now shift to a real demonstration of how useful functions are. We will use the Titannic dataset that we loaded into a dataframe earlier. Let's for a minute see what we can do with this dataset WITHOUT a function.

Let's we want to concatenate together some columns. So we do so for a few rows like so.

[18]:
```
concat1 = data['Name'][1] + ' | ' + data['Sex'][1] + ' | ' +␣
 ↪str(data['Age'][1]) + ' years old. | '
concat1
```

[18]:
```
'Wilkes, Mrs. James (Ellen Needs) | female | 47.0 years old. | '
```

[19]:
```
concat2 = data['Name'][2] + ' | ' + data['Sex'][2] + ' | ' +␣
↪str(data['Age'][2]) + ' years old. | '
concat2
```

[19]:
```
'Myles, Mr. Thomas Francis | male | 62.0 years old. | '
```

[20]:
```
concat3 = data['Name'][3] + ' | ' + data['Sex'][3] + ' | ' +␣
↪str(data['Age'][3]) + ' years old. | '
concat3
```

[20]:
```
'Wirz, Mr. Albert | male | 27.0 years old. | '
```

As we can see, while this is certainly doable, it is slightly tedious in nature. We have repeated our code three times manually to produce the same result. Let's see if we can arrive at the same result with a function.

[21]:
```
def concatColumns (row):
    '''
    This function concatenates several rows together into a sting.

    Parameters:
        row = The row in the dataframe that is being concatenated.
    '''


    string = row[4] + ' | ' + row[5] + ' | ' + str(row[6]) + ' years old.'

    print(string)
```

Now that we have our process canned into a function, let's see what it can do.

For demonstration purposes, we will take a small subset of the data, say 20 rows. We will use a for loop on the dataframe and process each row using our function.

[22]:
```
subset = data.sample(n = 20)
```

[23]:
```
subset
```

[23]:

	PassengerId	Survived	Pclass	Name	\
312	1204	0	3	Sadowitz, Mr. Harry	
377	1269	0	2	Cotterill, Mr. Henry Harry""	
62	954	0	3	Bjorklund, Mr. Ernst Herbert	
198	1090	0	2	Baimbrigge, Mr. Charles Robert	
21	913	0	3	Olsen, Master. Artur Karl	
228	1120	0	3	Everett, Mr. Thomas James	
266	1158	0	1	Chisholm, Mr. Roderick Robert Crispin	
315	1207	1	3	Hagardon, Miss. Kate	
110	1002	0	2	Stanton, Mr. Samuel Ward	

106	998	0	3	Buckley, Mr. Daniel
199	1091	1	3	Rasmussen, Mrs. (Lena Jacobsen Solvang)
161	1053	0	3	Touma, Master. Georges Youssef
145	1037	0	3	Vander Planke, Mr. Julius
336	1228	0	2	de Brito, Mr. Jose Joaquim
186	1078	1	2	Phillips, Miss. Alice Frances Louisa
291	1183	1	3	Daly, Miss. Margaret Marcella Maggie""
54	946	0	2	Mangiavacchi, Mr. Serafino Emilio
6	898	1	3	Connolly, Miss. Kate
112	1004	1	1	Evans, Miss. Edith Corse
121	1013	0	3	Kiernan, Mr. John

	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
312	male	NaN	0	0	LP 1588	7.5750	NaN	S
377	male	21.0	0	0	29107	11.5000	NaN	S
62	male	18.0	0	0	347090	7.7500	NaN	S
198	male	23.0	0	0	C.A. 31030	10.5000	NaN	S
21	male	9.0	0	1	C 17368	3.1708	NaN	S
228	male	40.5	0	0	C.A. 6212	15.1000	NaN	S
266	male	NaN	0	0	112051	0.0000	NaN	S
315	female	17.0	0	0	AQ/3. 30631	7.7333	NaN	Q
110	male	41.0	0	0	237734	15.0458	NaN	C
106	male	21.0	0	0	330920	7.8208	NaN	Q
199	female	NaN	0	0	65305	8.1125	NaN	S
161	male	7.0	1	1	2650	15.2458	NaN	C
145	male	31.0	3	0	345763	18.0000	NaN	S
336	male	32.0	0	0	244360	13.0000	NaN	S
186	female	21.0	0	1	S.O./P.P. 2	21.0000	NaN	S
291	female	30.0	0	0	382650	6.9500	NaN	Q
54	male	NaN	0	0	SC/A.3 2861	15.5792	NaN	C
6	female	30.0	0	0	330972	7.6292	NaN	Q
112	female	36.0	0	0	PC 17531	31.6792	A29	C
121	male	NaN	1	0	367227	7.7500	NaN	Q

[24]:
```
for row in subset.itertuples():

    concatColumns(row)
```

```
Sadowitz, Mr. Harry | male | nan years old.
Cotterill, Mr. Henry Harry"" | male | 21.0 years old.
Bjorklund, Mr. Ernst Herbert | male | 18.0 years old.
Baimbrigge, Mr. Charles Robert | male | 23.0 years old.
Olsen, Master. Artur Karl | male | 9.0 years old.
Everett, Mr. Thomas James | male | 40.5 years old.
Chisholm, Mr. Roderick Robert Crispin | male | nan years old.
Hagardon, Miss. Kate | female | 17.0 years old.
Stanton, Mr. Samuel Ward | male | 41.0 years old.
```

```
Buckley, Mr. Daniel | male | 21.0 years old.
Rasmussen, Mrs. (Lena Jacobsen Solvang) | female | nan years old.
Touma, Master. Georges Youssef | male | 7.0 years old.
Vander Planke, Mr. Julius | male | 31.0 years old.
de Brito, Mr. Jose Joaquim | male | 32.0 years old.
Phillips, Miss. Alice Frances Louisa | female | 21.0 years old.
Daly, Miss. Margaret Marcella Maggie"" | female | 30.0 years old.
Mangiavacchi, Mr. Serafino Emilio | male | nan years old.
Connolly, Miss. Kate | female | 30.0 years old.
Evans, Miss. Edith Corse | female | 36.0 years old.
Kiernan, Mr. John | male | nan years old.
```

So we can see how useful a function can be.

We can also call functions from within functions. Let's modify our function a bit to make the names all caps.

[25]:
```
def concatColumns2 (row):
    '''
    This function concatenates several rows together into a sting.

    Parameters:
        row = The row in the dataframe that is being concatenated.
    '''


    string = functionName(row[4]) + ' | ' + row[5] + ' | ' + str(row[6]) + '␣
↪years old.'

    print(string)
```

[26]:
```
for row in subset.itertuples():

    concatColumns2(row)
```

```
Converting Sadowitz, Mr. Harry to all caps!
SADOWITZ, MR. HARRY | male | nan years old.
Converting Cotterill, Mr. Henry Harry"" to all caps!
COTTERILL, MR. HENRY HARRY"" | male | 21.0 years old.
Converting Bjorklund, Mr. Ernst Herbert to all caps!
BJORKLUND, MR. ERNST HERBERT | male | 18.0 years old.
Converting Baimbrigge, Mr. Charles Robert to all caps!
BAIMBRIGGE, MR. CHARLES ROBERT | male | 23.0 years old.
Converting Olsen, Master. Artur Karl to all caps!
OLSEN, MASTER. ARTUR KARL | male | 9.0 years old.
Converting Everett, Mr. Thomas James to all caps!
EVERETT, MR. THOMAS JAMES | male | 40.5 years old.
Converting Chisholm, Mr. Roderick Robert Crispin to all caps!
CHISHOLM, MR. RODERICK ROBERT CRISPIN | male | nan years old.
```

```
Converting Hagardon, Miss. Kate to all caps!
HAGARDON, MISS. KATE | female | 17.0 years old.
Converting Stanton, Mr. Samuel Ward to all caps!
STANTON, MR. SAMUEL WARD | male | 41.0 years old.
Converting Buckley, Mr. Daniel to all caps!
BUCKLEY, MR. DANIEL | male | 21.0 years old.
Converting Rasmussen, Mrs. (Lena Jacobsen Solvang) to all caps!
RASMUSSEN, MRS. (LENA JACOBSEN SOLVANG) | female | nan years old.
Converting Touma, Master. Georges Youssef to all caps!
TOUMA, MASTER. GEORGES YOUSSEF | male | 7.0 years old.
Converting Vander Planke, Mr. Julius to all caps!
VANDER PLANKE, MR. JULIUS | male | 31.0 years old.
Converting de Brito, Mr. Jose Joaquim to all caps!
DE BRITO, MR. JOSE JOAQUIM | male | 32.0 years old.
Converting Phillips, Miss. Alice Frances Louisa to all caps!
PHILLIPS, MISS. ALICE FRANCES LOUISA | female | 21.0 years old.
Converting Daly, Miss. Margaret Marcella Maggie"" to all caps!
DALY, MISS. MARGARET MARCELLA MAGGIE"" | female | 30.0 years old.
Converting Mangiavacchi, Mr. Serafino Emilio to all caps!
MANGIAVACCHI, MR. SERAFINO EMILIO | male | nan years old.
Converting Connolly, Miss. Kate to all caps!
CONNOLLY, MISS. KATE | female | 30.0 years old.
Converting Evans, Miss. Edith Corse to all caps!
EVANS, MISS. EDITH CORSE | female | 36.0 years old.
Converting Kiernan, Mr. John to all caps!
KIERNAN, MR. JOHN | male | nan years old.
```

So we can see how the functions can ver very useful.