BIOS-584 Python Programming (Non-Bios Student)

Week 06

Instructor: Tianwen Ma, Ph.D.

Department of Biostatistics and Bioinformatics,
Rollins School of Public Health,
Emory University

Lecture Overview

- Functions in Python
  - Arguments, parameters, and return values
  - Define functions with def and return
  - Use functions to simplify the workflow
- Two-dimension Array (Matrix)

What is a function?

- A function is a block of code that performs a specific task
- To organize code, make it readable, and reusable
- If you perform the same task multiple times, you can write a function to do that task and call it whenever you need
  - If you do the same task >=3 times, you should use function to simplify
  - Similar to the purpose of for loop, i.e., to reduce the duplicated workload

What is a function?

- As your code grows, functions will help you keep it maintainable and scalable (especially for debugging)
- We have already seen functions in Python
  - print(), np.mean(), plt.hist(), isinstance(), etc.
- These functions are built-in, but you can customize your own functions to cater to your own needs.

Components of a function

- Parameters: input variables that a function expects to receive
- np.random.normal() expects two required parameters
  - Mean (loc)
  - Standard deviation (scale)
  - Sample size (size) is an optional parameter.

Components of a function

- Arguments and return values: The actual value each parameter takes and the final output from the function
- np.random.normal(loc=0, scale=1, size=None)
- It takes two arguments and returns a random number from a normal distribution with a mean of 0 and a standard deviation of 1.

Components of a function

- Default arguments: pre-specified values for certain parameters,
- Usually optional to change
- If you do not provide a value for the default argument, the function will use the default value
- How to tell whether it is required or optional?

Components of a function

- If you do not specify anything, it will generate one sample from the standard normal distribution.

- Go to their API page, if there is a value after each parameter, it is optional, otherwise, you have to specify to run the function successfully.
  - If you do not specify values to required parameters, Python will yield an error.

https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Enter arguments by assignment

- The most common way to pass arguments to a function is by assignment
- You can pass arguments by position or by name
  - By position: you cannot change the order of the arguments
  - By name: you can change the order of the arguments
- You can also use default arguments if you do not want to pass a specific value

Enter arguments by assignment

- To simulate one random sample from a uniform distribution from -2 to 2
- By position:
  - np.random.uniform(-2, 2, None)
  - Cannot change the order
- By name:
  - np.random.uniform(size=None, low=-2, high=2)
  - Okay to change the order b/c Python still recognizes the correspondence btw parameters and their arguments

numpy.random.uniform

`random.uniform(low=0.0, high=1.0, size=None)`

Draw samples from a uniform distribution.

https://numpy.org/doc/stable/reference/random/generated/numpy.random.uniform.html

Customize a function

- You can customize a function using def keyword
- General syntax:

```
#---- DEFINE
def my_function(parameter):
    body
    return expression

#---- RUN
my_function(parameter = argument)

#---- RUN
my_function(argument)
```

Customize a function

- Functional name should be descriptive
  - Usually separate by “_” for readability
- Parameters are variables that function expects to receive

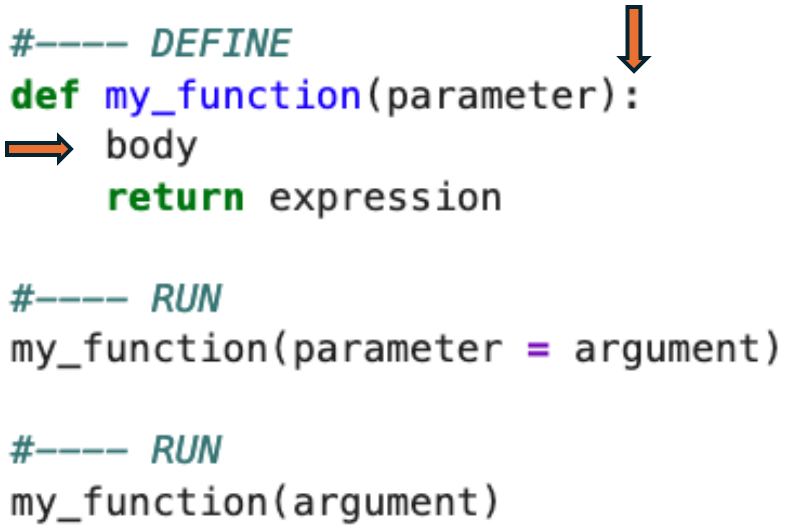
```
#---- DEFINE
def my_function(parameter):
    body
    return expression

#---- RUN
my_function(parameter = argument)

#---- RUN
my_function(argument)
```

Customize a function

- Body is the code to implement the task
  - ":" at the end of the first row
  - Indentation is required.
- Return is optional
  - If you don't provide a return statement, the function will return None
  - Good practice to always return something.
  - print("XXX is complete.")



Let's create a function!

- Recall we write a PMF function of Poisson distribution previously.
- Let's wrap it up with a function so that you can call it for future work if possible.
- The PMF of Poisson is mathematically defined as
- $P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword
- What are the parameters in this function?

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword
- What are the parameters in this function?
- Complete the body section of the function

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword
- What are the parameters in this function?
- Complete the body section of the function
- Assign the value of PMF to a variable pmf_val

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword
- What are the parameters in this function?
- Complete the body section of the function
- Assign the value of PMF to a variable pmf_val
- Finish the return statement (required)

Let's create a function!

$$P(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}, k \in \mathbb{N}_0$$

- Give a name "fun_poisson_pmf" with a def keyword
- What are the parameters in this function?
- Complete the body section of the function
- Assign the value of PMF to a variable pmf_val
- Finish the return statement (required)
- Run the function and call it in the next code chunk

Let's create a function!

- Suppose we let lambda_val be a required parameter and k be an optional parameter
- In the def row, write required parameter first followed by optional parameter then.
  - If you did it in the other way, Python will give you an error.
- For optional parameter, give it a default argument, i.e., k=1.

Ways to call the function

- By name but follow the original order
- By position
- By name but follow a random order

Example 2

- Define a new function to calculate the value of the following quadratic function

$$f(x) = x^2 + 2x + 1$$

- Test your function with $x = 2, 3$.

Example 3

- Call back the for loop from the last course
- Write two functions to simplify the hierarchical structure
- See details on the Jupyter notebook

The lambda function

- You may be curious why lambda is a reserved keyword.
- Lambda functions are short functions, which you can write in one line
- They can have any number of arguments but only one expression (no return statement)
- They are used when you need a simple function for a short period of time

The lambda function

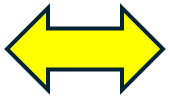
- They are also known as anonymous functions, although you can assign them to a variable
- Syntax:
  - my_function = lambda parameters: expression
- More information can be found here.

Example: Calculate $x + y$

```
def fn_sum(x, y):
    return x + y
```

```
fn_sum(1,2)
```

```
3
```



```
# (a) Define function
fn_sum = lambda x,y: x + y

# (b) Run function
fn_sum(1,2)
```

```
3
```

The lambda function

- Rewrite the PMF of a Poisson distribution using a lambda function (Example 4)

Example 5

- Boolean + Functions
- Write a function called fn_iseligible_vote
- This function returns a Boolean value that checks whether input age is greater or equal to 18.

Functions for visualizations

- Returning a value is not required.
- You can customize a plot
  - You can use functions to store your favorite aesthetic

Example 6

- Define a function red_histogram
- Parameters include vec_x and title

Example 7

- Create a function that computes a red scatterplot named red_scatterplot
- It takes two parameters vec_x and vec_y
- When you call the functions, you can use **acceleration** and **weight** as the corresponding arguments.

Arrays

- We briefly talked about 1-dim array before.
- Array is a grid that contains values of the same type.
- It can be more than 1-dim.

Arrays

- The array holds and represents any regular data in a structured way.

```python
# 1-dim array
arr_1d = np.array([1,2,3], dtype=np.int64)
# dtype is optional: depending on your operation system,
# it is np.int32 or np.int64 by default.
print(arr_1d)

# 2-dim array
arr_2d = np.array([[1,2,3], [4,5,6]], dtype=np.int64)
print(arr_2d)

# 3-dim array
arr_3d = np.array([[[1,2,3],
                    [4,5,6]],
                   [[-1,-2,-3],
                    [-4,-5,-6]]], dtype=np.int64)
print(arr_3d)
```

```
[1 2 3]
[[1 2 3]
 [4 5 6]]
[[[ 1  2  3]
  [ 4  5  6]]

 [[-1 -2 -3]
  [-4 -5 -6]]]
```

Arrays (Take arr_2d as an example)

- Raw data (memory address)
- How to locate an element (shape and indexing)
- How to interpret an element (data type).

```
[[1 2 3]
 [4 5 6]]
```

```
1 # print out memory address
2 print(arr_2d.data)
3
4 # print out shape
5 print(arr_2d.shape)
6
7 # print out data type
8 print(arr_2d.dtype)
```

```
<memory at 0x119532cf0>
(2, 3)
int64
```

Arrays (Take arr_2d as an example)

- Raw data (memory address)
- How to locate an element (shape and indexing)
- How to interpret an element (data type).

```
[[1 2 3]
 [4 5 6]]
```

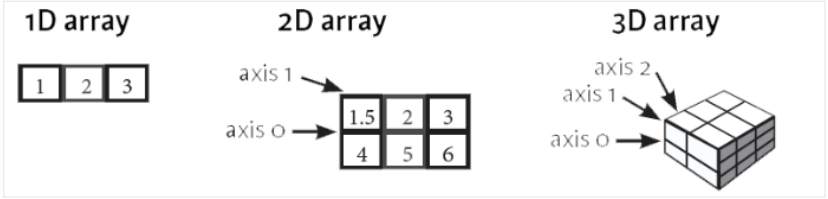
```python
# print out memory address
print(arr_2d.data)

# print out shape
print(arr_2d.shape)

# print out data type
print(arr_2d.dtype)
```

```
<memory at 0x119532cf0>
(2, 3)
int64
```

For a 2d-array, the shape is a coordinate of two integers (tuple). Axis 0 corresponds to # of rows, while axis 1 corresponds to # of columns.



How to initialize an np array

- Useful functions
- np.ones()
- np.zeros()
- np.random.random()
- np.empty()
- np.full()
- np.arange()
- np.linspace()

How to load np arrays from text

- np.loadtxt()
- np.genfromtxt()

How to save np arrays

- np.savetxt()
- np.save()
- np.savez()
- np.savez_compressed()