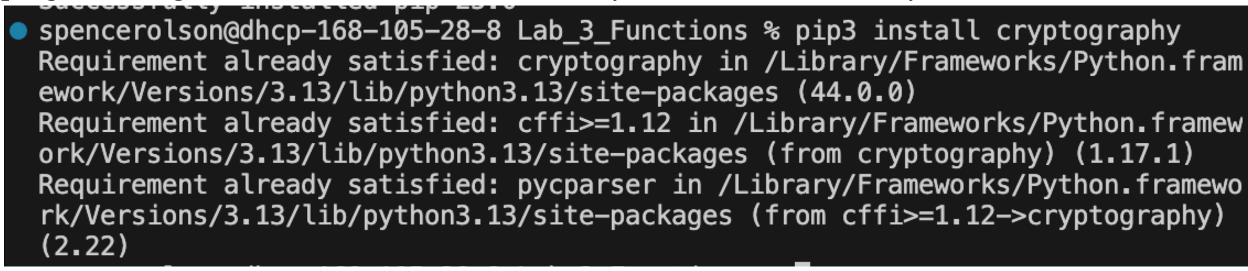
Spencer Olson February 9
Lab 3Functions

Lab for Chapter 2: Functions and Modules

1. Installing and using function libraries (modules).
    a. See if the `cryptography` library is installed using `pip show cryptography`. If not, `pip install cryptography`. Write a small program using `import` that tests that the library was installed correctly.

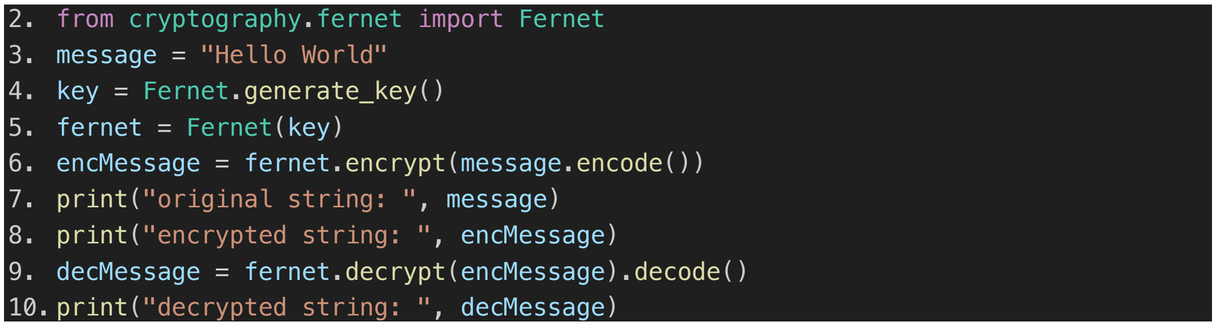


```
● spencerolson@dhcp-168-105-28-8 Lab_3_Functions % pip3 install cryptography
  Requirement already satisfied: cryptography in /Library/Frameworks/Python.fram
  ework/Versions/3.13/lib/python3.13/site-packages (44.0.0)
  Requirement already satisfied: cffi>=1.12 in /Library/Frameworks/Python.framew
  ork/Versions/3.13/lib/python3.13/site-packages (from cryptography) (1.17.1)
  Requirement already satisfied: pycparser in /Library/Frameworks/Python.framewo
  rk/Versions/3.13/lib/python3.13/site-packages (from cffi>=1.12->cryptography)
  (2.22)
```

i. Put your code and results here:

-In terminal we can input **pip3 show cryptography.**
-In python use **import cryptography**

b. Using online documentation, find and read the documentation for an appropriate function that might be used to encrypt then decrypt a string. Change your import to specifically load the library with this function (hint: use `from cryptography.xxx import xxx` where `xxx` is the library with the encrypt function. Use the function to encrypt and decrypt a string input from the user. Note that the string will need to be "encoded" before encrypting it and decoded after decryption.
    i. Code and results here and explain why the string needs to be encoded and decoded:



```python
from cryptography.fernet import Fernet
message = "Hello World"
key = Fernet.generate_key()
fernet = Fernet(key)
encMessage = fernet.encrypt(message.encode())
print("original string: ", message)
print("encrypted string: ", encMessage)
decMessage = fernet.decrypt(encMessage).decode()
print("decrypted string: ", decMessage)
```

c. How many parameters does the encryption function take? How do you know what the parameters are and what values they expect? Is it necessary to place the parameters in a certain order? Why/why not?

It takes one parameter as we have written unless it was written as an optional parameter, it returns a single data type in bytes and then re-decodes the single string (parameter) we provided. No certain order is required. Because it's a singly parameter.

a. Is the function named appropriately? Explain why or why not.

I would say yes because someone that you're working with or explaining the program to may not understand what code is. To ensure we know everyone will be able to understand what it is the code is doing. The identifier encrypt is a solid choice. You could expand on the name as far as the type of encryption however that's only needed in niche needs for security reasons.

b. Explain why it is not necessary to give the string as a parameter to the encode/decode functions. How does this work and why this is better than putting the string directly in the function call? Why is this different than the encryption/decryption functions where you do have to put the string in as an argument?

The return from input function is a string that we stored in message(). We generally would never want to encrypt anything other than a string. Therefore the function encrypt just encodes the message which in and of itself is a string.
We don't provide the string because it doesn't require us to define it further out of redundancies. It's better than putting it in the function call cause it ensuring we know what we are encoding cause its inside the string itself. It's different from the encrypt and decrypt functions because they are not something that is always needed for a string. It would also be silly to use it if we wanted to customize an encryption ourselves so that it isn't a well-known package in the world.

2. Create a function—call it *midpoint* —that takes two numbers as input and returns the value halfway between them.
   a. Code here:

```python
def midpoint(begin_value, end_value):
    mid = (begin_value+end_value)/2
    print('Test numbers', begin_value, end_value, "Midpoint is ", mid)
```

11. Explain the benefit of creating this function:
    Also known as a stub function, it's a stand in to assist in making larger problem sets. It will enable us to easily recall the function to return the

appropriate value within larger problems. It allows us to create arguments within a function to expand on the possibilities for future use. It states what we are doing as well.

12. Explain if the function is named appropriately:
    Yes, it is simple and to the point of the function's intent. Easy to remember etc.

3. Create a function—call it *squareroot*—that takes a number and returns the squareroot of that number. Use the fact that the square root of n is n**0.5.

    c. Code here:

```python
def squareroot(number):
    return number**.5

print(squareroot(4))
```

b. Explain why we might create this function rather than use n**0.5 whenever we want the squareroot:
We can reuse it later. Since its highlighted its clearly defines and documents what we are doing rather than N.5 showing up without obvious visuals. It also allows us to modify what we want out of a function.**

Now that you have two functions, create a new "module", called *HandyMath.py*, and put your two functions into that module. Now add three more handy math functions:

a. *exponent*, which takes two numbers, a base and an exponent, and returns the value when you raise the base to the power of the exponent.

    i. Code here:

```python
def exponent(base, exponent):
    exp = base**exponent
    return exp
```

b. *max*, which takes two numbers as input and returns the value of the larger one. Use the two comparison expressions or a conditional expression (do not use an if-statement).

    i. Code here:

```python
def max(num1, num2):
    return num1 if num1 >= num2 else num2
```

c. *min*, which takes two numbers as input and returns the value of the smaller one. Use the two comparison expressions or a conditional expression (do not use an if-statement).
   i. Code here:

```
def min(num1, num2):
    return num1 if num1 <= num2 else num2
```

5. Create a new file, call it *Use_Module.py*, that imports HandyMath.py, asks for two numbers from a user, and prints out the midpoint of those numbers, the square root of the square of one number, the result when raising one number to the exponent of the other, and finally the max and min of the numbers. Use the Python f-string capability to format these strings.
   A. Code here:

```
import HandyMath

n1 = float(input("Enter number 1:"))
n2 = float(input("Enter number 2:"))

print(f"The Midpoint of {n1} and {n2} is {HandyMath.midpoint(n1,n2)}")

```

   c. Explain the benefits of creating and using this custom module:

6. Look up the math functions built into Python (https://www.w3schools.com/python/python_math.asp). Have you replicated any of those functions in HandyMath?

Yes all but midpoint.

   a. Look up the math module (https://docs.python.org/3/library/math.html). Have you replicated any of those functions in HandyMath?

Yes , min max and pow

   b. Change your import to `from HandyMath import max, min` and rewrite and test your code. Explain how you are able to use HandyMath module functions in place of the built-in and math module functions. Also, explain how you know which function is being used.
      i. **You are calling print to execute the function you defined in handymath**

7. (Extra Credit)

c. Add a function to your HandyMath module that takes two numbers x,y and a function name as arguments then returns a string “The function <function name> x,y = <function applied to x,y>”. You can use `.__name__` to get the identifier of a variable. Try this out for min, max, and exponent.

   i. Code here:

```python
def arguments(x, y, func):
    return  f"The function {func.__name__}({x}, {y}) = {func(x, y)}"

```

a. What does this tell you about functions in Python?

They are very versatile and are able to have a hierarchy for ordering complex arguments.

b. Is there any benefit to passing functions as arguments to another function? Can you think of an example where this would be useful? Hint: Why might I want to give a callback function for a function rather than return a value?

Yes, same as before with flexibility, customization and reusability. Using a callback function is useful for asynchronous event driven systems.