1 Tasks

1.1 print vs return - print_vs_return.py

```
# print_vs_return.py
def f1(x):
    print(x + 1)

def f2(x):
    return x + 1

# Example 1:
f1(3)
# 4

f2(3)
# nothing happens
\\
\\
# Example 2:
a = f1(3)
#see a printed 4
b = f2(3)
#see nothing
a
# we get nothing!
b
# NOW we see 4
```

What are the values of a and b? Why are they different?
a has no value, but b has been bound to 4

Example 3: Now run the following code:

```
# Example 3:
print(f1(3))
# prints 4 and then None, showing how we called the function and then printed a none-type
print(f2(3))
# prints only 4, showing how we printed a saved value of 4

# Example 4:
f1(3) + 1
# 4, then unsupported operand type because we can't add nonetype to 1!
f2(3) + 1
# 5
```

1.2 Mystery program - mystery.py

```python
def b(z):
    prod = a(z, z)
    print(z, prod)
    return prod

def a(x, y):
    x = x + 1
    prod = x * y
    return prod

def c(x, y, z):
    sum = x + y + z
    pow = b(sum) ** 2
    return pow

x = 1
y = x + 1
res = c(x, y+3, x+y)
print(res)
```

When we run this code, the first function we call is c(x, y+3, x+y). We start by evaluating the arguments: x = 1, and y = x + 1 = 1 + 1 = 2. This means we run c(1, 5, 3). The first line of function c is finding the sum: 1+5+3 = 9. Then we return b(sum) ** 2. When we call b(9), our first line is binding the result of a(9, 9) to 'prod'. When we call function a, we pass 9 as x and 9 as y. x is rewritten to 1+9 = 10. prod is then equal to 10*9 = 90, and we return 90. We bind 90 to prod in function b (which is a different prod than the one in function a), and then print z, prod, which is still 9, 90 (note the ',' doesn't actually print: we can just pass many arguments to print which will be separated by a space). After that, we can finally go back to function c. b(sum) was equal to 90, so pow = 90**2 (90^2) which is 8100. When we print that result, our final screen will look like:

```
9 90
8100
```

This is clearly a lot of text, and don't worry if you still don't see it!! Just remember that the computer is Perfectly Logical (which to a coder, can be read as Perfectly Annoying), so you can just read line by line! When one line sends you to a new point/funtion, we just read from then until we finish the 'side-task', and go back to the main code.

1.3 Not equal - not_equal.py

```python
#sample not.equal.py
def not_equal(a, b):
    is_equal = a == b
    return not is_equal

#testing section
a = 12
b = 12
c = 2
d = 9
not_equal_test1 = not_equal(a, b)
print('A =', a)
print('B =', b)
print(not_equal_test1)\textbf{}
not_equal_test2 = not_equal(c, d)
print('C =', c)
print('D =', d)
print(not_equal_test2)
```

1.4 Making desserts with Python - desserts.py

(a)

```python
# This program prints out recipes for desserts.
def tiramisu():
    print('Tiramisu (Wolfgang Puck)')
    print()
    ladyfingers()
    print()
    mascarpone_cream()
    print()
    espresso_syrup()
def ladyfingers():
    print('Ladyfingers: ')
    print('* 6 eggs, separated')
    print('* 1 cup cake flour, sifted')
    print('* Melted butter, for brushing pan')
def mascarpone_cream():
    print('Mascarpone Cream: ')
    print('* 6 egg yolks')
    print('* 1 cup sugar')
    print('* 1/3 cup Marsala')
    print('* 1/4 cup brandy')
    print('* 2 pounds mascarpone cheese')
def espresso_syrup():
    print('Espresso Syrup: ')
    print('* 1 cup espresso, hot')
    print('* 3 tablespoons brown sugar')
    print('* 1 tablespoon sugar')
    print('* 1 teaspoon lemon juice')
    print('* 1 teaspoon vanilla extract')
    print('* 1/2 cup grated bittersweet chocolate')
```

(b)

```python
#because of our nice code structure, this is straightforward!!
#simply add this function
def snazzy_border():
    print('<><><><><><><><>[][][][][][][][]<><><><><><><><><>')
#then update the 'tiramisu' function thusly:
def tiramisu():
    snazzy_border()
    print('Tiramisu (Wolfgang Puck)')
    snazzy_border()
    ladyfingers()
    print()
    mascarpone_cream()
    print()
    espresso_syrup()
    snazzy_border()
```

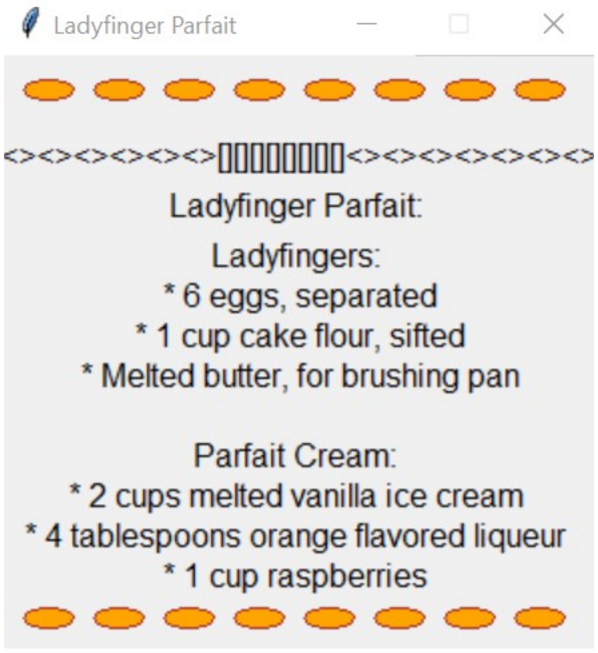
(c)

```python
#adding a new method for parfait cream
def parfait_cream():
    print('Parfait Cream: ')
    print('* 2 cups melted vanilla ice cream ')
    print('* 4 tablespoons orange flavored liqueur ')
    print('* 1 cup raspberries ')

#function to print the whole recipe
def lady_parfait():
    snazzy_border()
    print('Ladyfinger Parfaits: ')
    snazzy_border()
    parfait_cream()
    print()
    ladyfingers()
    snazzy_border()

#calls to print both recipes
tiramisu()
print()
lady_parfait()
```

2 Optional Challenge[1]:

2.1 Yummy graphics - parfait_graphics.py

Have fun with this!! Here's an example from one of the tutors!



```
<><><><><><>▯▯▯▯▯▯▯▯▯<><><><><><><>
Ladyfinger Parfait:
Ladyfingers:
* 6 eggs, separated
* 1 cup cake flour, sifted
* Melted butter, for brushing pan

Parfait Cream:
* 2 cups melted vanilla ice cream
* 4 tablespoons orange flavored liqueur
* 1 cup raspberries
```

[1]optional problems are provided for those who have further interests and want to explore more. You are not responsible for those questions, however.