

1. Warming up

- (a) An object is made up of two main components, what are they?

Objects have data AND the methods that operate on that data, called using the dot notation '.'

- (b) The function below *could* enter an infinite loop and never finish depending on the value the function is called with. State what conditions would cause it to enter an infinite loop assuming `num` is a number.

```
def first(num):  
    while num != 0:  
        print(num)  
        num = num - 1
```

`num < 0`

- (c) If we had the following program in a file and we “ran” the program using the green arrow in Spyder, what would you see displayed?

```
def mystery(a_string):  
    return a_string + a_string  
  
def mystery2(a_string):  
    print(a_string * 2)  
  
mystery("CS150 is good ")  
mystery2("CS150 is great ")
```

**The first call doesn't print anything since the function just returns a value!
The second line prints: CS150 is great CS150 is great**

- (d) DNA is made up of four different bases, abbreviated A, C, G and T. A DNA sequence consists of some combination of these letters. Write a function called `random_DNA` that generates a random DNA sequence. Your function should take a single parameter that is the length of the sequence to generate.

```
from random import randint

def random_DNA(length):
    sequence = ""

    for i in range(length):
        pick = randint(1,4)

        if pick == 1:
            sequence += "A"
        elif pick == 2:
            sequence += "C"
        elif pick == 3:
            sequence += "G"
        else:
            sequence += "T"

    return sequence
```

- (e) What values are contained in the variables `a`, `b`, and `c`, after the following 6 statements are executed? For partial credit, also list the values after the first 3 statements.

```
a = 10
b = [a, 20]
c = a * a
a = b[1]
b = c // 3
c -= a
```

a: 20
b: 33
c: 80

2. Turtle fun

Draw below what the following program would draw on the screen assuming the screen width and height are 700 (-350 to 350). For clarity, feel free to include lines marking the x and y axes. The `circle` function draws a circle starting the very bottom of the circle.

```
from turtle import *

X_MIN = -300 # the smallest x value on the screen
X_MAX = 300 # the largest x value on the screen

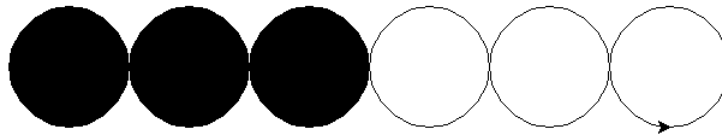
def draw_pattern(radius):
    x = X_MIN

    while x < X_MAX:
        pu()
        goto(x, 0)
        pd()

        if x < 0:
            begin_fill()
            circle(radius)
            end_fill()
        else:
            circle(radius)

        x += 2*radius

draw_pattern(50)
done()
```



3. Strings and Lists

- (a) Write a Python function `stringToIntList` that takes a string representing a number and returns a list of ints representing the individual digits. For example:

```
>>> stringToIntList("80345")
[8, 0, 3, 4, 5]
```

```
def stringToIntList(string):
    list = []

    for i in range(len(string)):
        list.append(int(string[i]))

    return list
```

- (b) Suppose that `fruits` is initialized as follows:

```
fruits = ['kiwi', 'banana', 'papaya', 'mango', 'melon']
```

What would the following statements print (make sure to be clear what the type of the thing is, i.e. string vs. list):

```
- print(fruits[1])
  banana
- print(fruits[:1])
  ['kiwi']
- print(fruits[-3:-1])
  ['papaya', 'mango']
- print(fruits[2][4])
  y
- print(fruits[1:3] * 2)
  ['banana', 'papaya', 'banana', 'papaya']
- print(fruits[1] * 2)
  bananabanana
```

- (c) What would the following function return if called with “computer science” as the argument (i.e. `hmm("computer science")`):

```
def hmm(phrase):
    output = ""

    for i in range(len(phrase)):
        if output.find(phrase[i]) == -1:
            output = output + phrase[i]

    return output
```

computer sin

4. Conditionals

- (a) Two of the following three functions do the same thing. Identify the two that are the same:

```
def a(x, y):
    if (x < 0 and y < 0) or (x > 0 and y > 0):
        return x * y
    else:
        return 0
```

```
def b(x, y):
    if x < 0 and y < 0:
        return x * y
    elif x < 0:
        return 0
    else:
        return x * y
```

```
def c(x, y):
    if x < 0 and y < 0:
        return x * y
    elif x > 0 and y > 0:
        return x * y
    else:
        return 0
```

First and third

- (b) The following function checks to see whether two numbers are within 10 of each other. The function works correctly, but could be written much more elegantly. Rewrite the function to be more elegant.

```
def within_10(a, b):
    if abs(a - b) <= 10:
        return True
    elif abs(a - b) > 10:
        return False
```

solution:

```
def within_10(a, b):
    return abs(a - b) <= 10
```

5. File processing

Write a function called `every_other_line` that takes as a parameter a filename and reads *every other line* from the file into a list and returns that list. The function should start with the first line, that is add the first line, the third line, the fifth line, etc into the list

```
def every_other_line(filename):
    file = open(filename, "r")
    data = []
    count = 0

    for line in file:
        if count % 2 == 0:
            data.append(line.strip())

        count += 1

    return data
```