 Python Programming

Intermediate Python: Practice Problems

In this homework, you're going to write code for a few problems. We'll be building on older material like lists, loops, and functions, as well as adding in some practice from the new materials.

You will practice these programming concepts we've covered in class: * Using functions with parameters. * Using dictionaries and sets. * Declaring and using classes to make objects. * Demonstrating inheritance and method overriding. * Using a variable number of keyword arguments.

Deliverables

For each of the challenges listed below, you will create a new `.py` file and write code to solve the problem. For example, you would create `problem1.py` with your solution code to the first problem. Run the file from the command line to check your work.

Reminder: On your laptop, you can run the file from your command line with the following:

```
python3 problem1.py
```

> **Hint:** Make sure you are printing something out with the `print` statement. Otherwise, you won't see any output from running your program!

Requirements:

- By the end of this, you should have six different `.py` files (one for each problem).

Homework Problems

Problem 1: Call Me! (Maybe.)

Skills you're practicing: Accessing dictionaries, writing functions, and passing parameters.

Write a function called `get_contact()` that has two parameters. The first will be a dictionary called

`contacts` and the second will be a string called `name` for which you would like to look up the phone number.

Example Test Code

```
contacts = {
  "Carly": "333-3333",
  "Blondie": "444-4444",
  "Jenny": "867-5309"
}
name = "Jenny"

phone_number = get_contact(contacts, name)

print("The phone number of", name, "is", phone_number)
```

Example test output:

```
The phone number of Jenny is 867-5309
```

Hint 1:

Refer to your class notes for details about how to use dictionaries.

Hint 2:

As a reminder, let's walk through an example of creating, assigning to, and checking if a value is in a dictionary.

```
# Creating a dictionary called my_dict:
my_dict = {}

# Assigning key-value pairs to my_dict:
my_dict["foo"] = 1

# Checking if my_dict contains the key "bar:"
if "bar" in my_dict:
    print("bar is here")
else:
    print("bar is not here") # This is what prints.
```

Hint 3:

Careful! Python requires that you insert a key into a dictionary before you try to modify its value. If you try to access a dictionary at a key that hasn't been added, you'll get an error and the program will crash. Remember to use an `if` statement to see if a key is *in* a dictionary before you try to read it!

```
d2 = {}
d2["foo"]
# KeyError: 'foo'
```

Problem 2: Hey Jude, Don't Make Me Count!

Skills you're practicing: Using dictionaries and writing a function with a parameter.

Write a function called `letter_counter()` that returns a dictionary called `counts`, which has a character as the key and a count as a value. The goal is to count up the number of occurrences of each letter in a string.

Here is how your function will be called, and what is expected as the output.

Example test code:

```
word_to_count = "banana"

result = letter_counter(word_to_count)

print(result)
```

Example test output:

```
{'b': 1, 'a': 3, 'n': 2}
```

Hint 1:

Remember that you can iterate over a string one letter at a time using a `for` loop.

```
for letter in "alpha":
    print(letter)
```

Hint 2:

Remember to write a return statement at the end of your function. No code will execute after the return statement because the return statement also exits the function. What you return from the function will be assigned to the `result` variable in the example above.

Hint 3:

Think you have the answer? To really test your function, try putting in a whole song as your string! Here are the lyrics to "Hey Jude" already written into a Python string. Here's the expected output:

```
{'H': 4, 'e': 143, 'y': 45, ' ': 368, 'J': 23, 'u': 53, 'd': 52, ',': 44, 'o': 69,
'n': 191, "'": 11, 't': 84, 'm': 23, 'a': 205, 'k': 15, 'i': 33, 'b': 19, '.': 20,
'T': 7, 's': 18, 'g': 11, 'r': 49, 'R': 3, 'l': 24, 'h': 208, 'c': 5, 'f': 7, 'Y':
3, 'w': 11, 'A': 2, 'p': 4, 'D': 1, 'F': 1, 'B': 2, 'N': 18, 'v': 2, 'S': 1, 'j':
1}
```

Problem 3: Kingdom, Phylum, CLASS!

Skills you're practicing: Making classes and creating instances.

Let's create a class. Remember a class is a pattern for an object. In this case, we will create an `Animal` class that will be able to generate any number of animal instances with the same properties.

- Create an `Animal` class.
  - When an animal is created, print `I am coming to life!`.
  - Animals keep track of their current `energy` as an integer (this is assigned to the `self`).
  - Animals have an `eat()` method, which takes one numerical argument, `amount`, and increases the `energy` by `amount`.
  - Animals have a `move()` method, which depletes the `energy` by 10 and prints the statement `I am running!`.
  - Animals have a `get_status()` method, which prints the current `energy` level as well as a message about their energy state. If the `energy` level is:
  - Below `0`: Print `I'm starving!`.
  - Between `0` and `50`: Print `I'm getting hungry!`.
  - Between `50` and `100`: Print `I'm happily full.`.
  - Above `100`: Print `I'm feeling stuffed!`.
  - Animals are created with a default `energy` of `50`.
  - Animals have a `say_hi()` method that prints `Meep!`.

Example test code:

```
print("Making my first animal")
first_animal = Animal()
first_animal.get_status()
first_animal.eat(55)
first_animal.get_status()
first_animal.move()
first_animal.get_status()
print("first animal saying hi")
first_animal.say_hi()
print()

print("Making a second animal now")
second_animal = Animal()
second_animal.get_status()
second_animal.eat(-45)
```

```
second_animal.get_status()
second_animal.move()
second_animal.get_status()
print("second animal saying hi")
second_animal.say_hi()
```

Example test output:

```
Making my first animal
I'm coming to life!
My energy level is 50
I'm happily full.
My energy level is 105
I'm feeling stuffed!
I am running!
My energy level is 95
I'm happily full.
first animal saying hi
Meep!

Making a second animal now
I'm coming to life!
My energy level is 50
I'm happily full.
My energy level is 5
I'm getting hungry!
I am running!
My energy level is -5
I'm starving!
second animal saying hi
Meep!
```

Hint:

Don't forget the constructor! This is the code that runs when we are creating an instance. The constructor in Python is `__init__`.

Problem 4: Fly Like an Eagle

Skill you're practicing: Using inheritance.

Let's practice writing classes and using inheritance by modeling different types of animals. Use your `Animal` class fas the parent class from which your new classes, `Penguin` and `Eagle`, will inherit.

- Create a `Penguin` class that **inherits** from `Animal`.
  - `Penguin`s are created with an `energy` level of `100`.
  - `Penguin`s shout, `I am a penguin!` when they are are created.
  - A `Penguin`'s `move()` method depletes the `energy` by `5` and prints `I am sliding!`.

- Create an `Eagle` class that **inherits** from `Animal`.
  - An `Eagle` begins its life with an `energy` level of `20`.
  - `Eagle`s shout, `I am an eagle!` when they are are created.
  - An `Eagle`'s `move()` method depletes the `energy` by `20` and prints `I am flying to the sea!`.
  - An `Eagle`s `say_hi()` method prints `shrieeeeeek!`.
  - `Eagle`s cannot `move` if their `energy` is less than `0`. Instead print `I'm too tired to fly...`.

Example test code:

```
animal = Animal()
animal.get_status()
animal.eat(60)
animal.get_status()
animal.move()
animal.get_status()
animal.say_hi()
print()

penguin = Penguin()
penguin.eat(5)
penguin.get_status()
penguin.move()
penguin.get_status()
penguin.say_hi()
print()

eagle = Eagle()
eagle.say_hi()
eagle.get_status()
eagle.move()
eagle.get_status()
eagle.move()
eagle.get_status()
eagle.move()
print()
```

Example test output:

```
I'm coming to life!
My energy level is 50
I'm happily full.
My energy level is 110
I'm feeling stuffed!
I am running!
My energy level is 100
I'm happily full.
Meep!

I'm coming to life!
I am a penguin!
My energy level is 105
I'm feeling stuffed!
I am sliding!
```

```
My energy level is 100
I'm happily full.
Meep!

I'm coming to life!
I am an eagle!
Shrieeeeek!
My energy level is 20
I'm getting hungry!
I am flying to the sea!
My energy level is 0
I'm getting hungry!
I am flying to the sea!
My energy level is -20
I'm starving!
I'm too tired to fly…
```

Hint 1:

Don't forget to call the parent's constructor! You can do this by calling `super().__init__()` in the child class' constructor.

Hint 2:

Remember, you only need to write a method in the child class if it is **different** than the one in the parent class. For example, the `Penguin` class does not need its own `say_hi()` method because we know our `Animal` class has a `say_hi()` method already that does exactly what we want it to do. On the contrary, with `Eagle`, we will have to override the `say_hi()` method in order to make it do what we want instead of the default behavior.

Problem 5: Ready, SET, Go!

Skill you're practicing: Using sets.

Let's say you own a restaurant. You ask the manager to take stock of the spices you have on hand so you know what you need to buy. Unfortunately, you get a list with a bunch of duplicates!

Convert this `spices_onhand` list to a set. Then, given the `spices_needed` set, print out what you need to buy.

You know that you need the following ingredients on hand at all times. It's a unique set of items, so you've stored this data in a `Set`.

```
spices_needed = set({"salt", "pepper", "ginger", "oregano", "paprika", "basil",
"curry powder", "cumin", "cayenne", "lemon pepper", "chili powder", "nutmeg",
"cinnamon", "star anise", "garlic salt", "coriander", "cardamom", "thyme"})
```

Example test input:

```
spices_onhand = ['cumin', 'nutmeg', 'salt', 'cumin', 'star anise', 'salt', 'basil',
'nutmeg', 'cumin', 'paprika', 'curry powder', 'pepper', 'curry powder', 'curry
powder', 'cayenne', 'cumin', 'star anise', 'star anise', 'curry powder', 'salt',
'salt', 'cardamom', 'cayenne', 'star anise', 'chili powder', 'curry powder',
'thyme', 'thyme', 'cayenne', 'nutmeg', 'basil', 'star anise', 'chili powder',
'oregano', 'coriander', 'nutmeg', 'chili powder', 'coriander', 'paprika', 'pepper',
'thyme', 'nutmeg', 'paprika', 'cayenne', 'basil', 'cinnamon', 'curry powder',
'cardamom', 'star anise', 'pepper', 'salt', 'curry powder', 'thyme', 'cardamom',
'salt', 'pepper', 'paprika', 'salt', 'cinnamon', 'cumin', 'curry powder',
'cardamom', 'cumin', 'cardamom', 'oregano', 'cardamom', 'pepper', 'star anise',
'pepper', 'cayenne', 'chili powder', 'cardamom', 'nutmeg', 'pepper', 'cardamom',
'curry powder', 'thyme', 'basil', 'nutmeg', 'coriander', 'paprika', 'curry powder',
'cayenne', 'cumin', 'nutmeg', 'paprika', 'star anise', 'thyme', 'curry powder',
'cardamom', 'oregano', 'basil', 'cinnamon', 'oregano', 'coriander', 'curry powder',
'cumin', 'thyme', 'pepper', 'thyme', 'cardamom', 'cayenne', 'chili powder', 'basil',
'pepper', 'cumin', 'thyme', 'cardamom', 'star anise', 'cayenne', 'cinnamon',
'cinnamon', 'cinnamon', 'cardamom', 'curry powder', 'curry powder', 'pepper',
'chili powder', 'pepper', 'cinnamon', 'cardamom', 'basil', 'thyme', 'cinnamon',
'cumin', 'nutmeg', 'cinnamon', 'cayenne', 'cardamom', 'nutmeg', 'cardamom',
'paprika', 'cumin', 'cayenne', 'chili powder', 'cinnamon', 'cumin', 'star anise',
'cardamom', 'thyme', 'basil', 'paprika', 'basil', 'oregano', 'cardamom', 'pepper',
'oregano', 'nutmeg', 'nutmeg', 'salt', 'basil', 'cayenne', 'oregano', 'star anise',
'star anise', 'oregano', 'salt', 'pepper', 'cinnamon', 'basil', 'salt', 'cardamom',
'cayenne', 'oregano', 'cinnamon', 'pepper', 'cumin', 'thyme', 'thyme', 'oregano',
'oregano', 'star anise', 'paprika', 'thyme', 'cinnamon', 'cinnamon', 'oregano',
'star anise', 'oregano', 'chili powder', 'cayenne', 'oregano', 'cumin', 'paprika',
'nutmeg', 'star anise', 'coriander', 'star anise', 'nutmeg', 'chili powder', 'star
anise', 'paprika', 'salt', 'salt', 'cayenne', 'curry powder', 'thyme', 'oregano',
'curry powder', 'curry powder']
```

Example test output:

```
Go shopping for: {'garlic salt', 'ginger', 'lemon pepper'}
```

Hint:

Look up set difference. This is basically subtracting the items of one set from another. If you had two sets — one of ingredients you need and one of ingredients you have — you could subtract the items you have from the ones you need in order to know what you will have to purchase.

Problem 6: Kwargosaurus

Skills you're practicing: Using variable number of keyword arguments (`**kwargs`) and writing functions.

The mighty Kwargosaurus stomps through the prehistoric Pythonic jungle, fighting with any dinosaur smaller than him. However, if a dinosaur his own age comes along, he cries and runs away! Write a function called `kwargosaurus()` that takes in a variable number of other dinosaurs for Kwargosaurus to encounter, with the key being the dinosaur name and the value being either `bigger` or `smaller`.

Example function call:

```
kwargosaurus(Velociraptor="smaller", Stegosaurus="smaller", Triceratops="smaller",
Trex="bigger")
```

Example output:

```
Velociraptor is small! Mighty Kwargosaurus will fight you!
Stegosaurus is small! Mighty Kwargosaurus will fight you!
Triceratops is small! Mighty Kwargosaurus will fight you!
Trex is big! Whimpering Kwargosaurus cries and runs away!
```

Hint: Remember that, for a variable number of arguments, you need a loop! You can access both the key and value of each parameter by using `kwargs.items()`. If you need syntax help with kwargs, check out this article.

Last, But Not Least

We threw more difficult problems your way than we did the last time, but you handled it! Shake it off and go relax!