Name:_____ **Partner:** _____

Python Activity 24b: Classes - Slots

Learning Objectives
Students will be able to:
Content:
- Explain how **slots** differ from attributes
- Explain why the slots attribute is useful

Process:
- Write code that creates a new user-defined class with slots

Prior Knowledge
- Python concepts from Activities 1-24.

Folks, this is a brand new activity. If you encounter any issues/typos, please let Iris know!

Critical Thinking Questions:

1. Examine the following code from interactive python below using a Flower data structure.

Interactive Python

```
0 >>> class Flower:
1 ...     """ A new class representing flowers """

2 >>> iris = Flower()
3 >>> iris.petals = 3
4 >>> iris.petals
5 3
6 >>> iris.bloomTime
7 AttributeError: 'Flowe' object has no attribute
'bloomTime'
```

a. What type of object is `iris`? How do you know?

_____

b. On which line is `iris.petals` on the lefthand side of an assignment operator?

What value is assigned? _____

c. On which line is `iris.bloomTime` on the lefthand side of an assignment operator?

_____

d. Why might `iris.bloomTime` on line 7 throw an error?

_____

e. Write a line of python to enter before line 6, to fix the error:

_____

2. Examine the following code below, which continues from the previous example:

```
8  >>> daisy = Flower()
9  >>> daisy.nonsense = 'wut WUT'
10 >>> daisy.nonsense
11 'wut WUT'
```

   a. What differs between our asisgnment of `daisy` in this example, and `iris` in the earlier example? _____

   b. Where do we assign a value to `daisy.petals` in this example? _____

   c. Where do we assign a value to `daisy.nonsense` in this example? What's its value?

   _____

   d. Is `nonsense` a meaninful attribute for objects of type `Flower`?

   _____

3. Examine the following code below, that overwrites previous versions of `Flower`:

Interactive Python

```
0 >>> class Flower:
1 ...     __slots__ = ['petals']

2 >>> rose = Flower()
3 >>> rose.petals = 5
4 >>> rose.nonsense = 'May'
5 AttributeError: Flower object has no attribute
'nonsense'
```

   a. How does the assignment of `rose.petals` differ from the assignment of `iris.petals` in question 1? _____

   b. How does the assignment of `rose.nonsense` differ from the assignment of `daisy.nonsense` in the previous question?

   _____

   b. What happens with line 5 in this example that didn't occur in the previous question?

   _____

   c. How does the definition of the `Flower` class differ in this example, from the definition of `Flower` used in questions 1-2?

   _____

   _____

FYI: The **__slots__** keyword defines a list of attributes for a class object. No additional attributes can be added to an instance, unless their name appears in the __slots__ list.

   d. What might happen if we modify line 1 to be `__slots__ = ['petals','nonsense']` and then ran the code?

_____

_____

4. Examine the following code below, which continues from the previous example:

```
6  >>> violet = Flower()
7  >>> violet.petals = 5
8  >>> violet.petals
9  5
10 rose.petals + violet.petals
11 10
```

   a. What is stored in `violet.petals`?

      _____

   b. What is happening on line 10?

      _____

5. Examine the following code below, which continues from the previous example:

```
12 >>> def avgPetals(flwrList):
13 ...     total = 0
14 ...     for flwr in flwrList:
15 ...         total += flwr.petals
16 ...     return total / len(flwrList)
```

   a. What is an example value for `flwrList`?

      _____

   b. What would the output for your example value in (a) result in?

      _____

   c. What does `avgPetals` do?

      _____

   d. Write a function, **`droughtPetals`**, that accepts a `Flower` object as a parameter and an integer `days`, and removes one petal from the flower for each `days` of drought:

      _____

      _____

      _____

      _____

      _____

      _____

Application Questions: Use Python to check your work

1a. Create a class, `Student`. Create an instance of `Student` which has a `name` and a `hobby` as instance slots.

1b. Create a function, `findAllHobbies` that takes a list of `Student` objects and returns a list of every student hobby.

1c. Create a function, `flipHobbies` that takes a list of `Student` objects and swaps the hobbies of the first & last person in the list, the second and second-to-last person in the list, the third and third-to-last person in the list, etc.

1d. Write a few lines of code for interactive python that uses all of the above functions you wrote: