Note: This is a new, untested POGIL. Let Iris know if there are issues!

Name:_____ **Partner:** _____

Python Activity 20: Import & Modules

Python has some built-in features for running code as a script vs. importing as a module.

Learning Objectives
Students will be able to:
Content:
- Explain the use of the **import** statement
- Predict the behavior of code when run as a script versus **module**

Process:
- Write code to import our own user-defined functions
- Write code whose behavior differs as a script and module

Prior Knowledge
- Expressions, variables, arithmetic, input

Critical Thinking Questions:

1. Closely examine the Python program below.

Python Program:

```
from math import pi

radius = int(input("Radius: "))
area = pi * radius**2
print(area)
```

a. What do you think the output might be, if the user enters a radius of 1? _____
b. What might the value of `pi` be? _____
c. Circle the new **keywords** in the code we haven't yet seen.
d. What might those new keywords do? _____
_____
_____
_____
_____

2. Examine the following code, it is very similar to the code that we explored in question 1.

```
from math import *

radius = int(input("Radius: "))
area_roundup = ceil(pi * radius**2)
print(area_roundup)
```

a. Underline where this new code differs from the previous example.
b. How might the output for this program compare with the output for the previous program?
_____

c. Is `ceil(..)` a variable, a function, a boolean, or a string? _____

d. Why might we have needed to change the text in the first line?
_____

Note: This is a new, untested POGIL. Let Iris know if there are issues!

FYI: **import** allows us to use code from external **modules** (or libraries), so we can make use of the definitions constructed in those modules. We can either import definitions one at a time with ***from <module name> import <definition name>*** or *all* the definitions in the module with ***from <module name> import ****.

3. Closely examine the Python program, area.py, below.

Python Program: area.py

```
from math import pi

def get_area(radius):
    area =  pi * radius**2
    return area
```

a. Trace through the program, how does the command flow differ in this example from the first example?

_____

b. When we run interactive python, here is the output we observe:

```
>>> from area import get_area
>>>
```

What happened?

_____

c. We can continue to use interactive python to write more code, as below:

```
>>> from area import get_area
>>> get_area(1)
3.141592653589793
```

What happened?

_____

d. What would we observe if we ran `python3 area.py` (run as a script)?

_____

e. How would we modify this code so it gives us meaningful output when we run it as a script?

_____

_____

_____

f. How does the *area* module differ from the *math* module?

_____

4. Below, we've added some code to area.py and saved it as area2.py.

Python Program: area2.py

```
from math import pi

def get_area(radius):
    area =  pi * radius**2
    return area

rad = int(input("Radius: "))
print(get_area(rad))
```

a. Trace through the program, how does the command flow differ in this example from the previous?

_____

b. When we run area.py as a script from the Terminal with `python3 area2.py`, here is the output:

```
python3 area2.py
Radius:
```

Note: This is a new, untested POGIL. Let Iris know if there are issues!

How does this output differ from the previous example (area.py)? Why might this be?

_____

_____

_____

c. When we import getArea from the area module, here is the output we observe:

```
>>> from area2 import get_area
Radius:
```

How does this output differ from the previous example (area.py)? Why might this be?

_____

_____

_____

d. Can you imagine a situation in which the user might want to use the `getArea(..)` function, but not have python prompt the user for an input radius?

_____

_____

_____

FYI: Sometimes, we want to import python code, and other times we may want to run it as a standalone script. To do this, python has a special keyword ***__name__***. If ***__name__*** is ***"__main__"***, then the code is being run as a script.

5. Below, we've added some code to area.py and saved it as area3.py.

Python Program: area3.py

```
from math import pi

def get_area(radius):
    area =  pi * radius**2
    return area

if __name__ == "__main__":
    rad = int(input("Radius: "))
    print(get_area(rad))
```

a. When we run area3.py as a script from the Terminal with `python3 area3.py`, here is the output:

```
python3 area3.py
Radius:
```

When we import `getArea` from the area module, here is the output we observe:

```
>>> from area2 import get_area
>>>
```

How do these outputs differ from the previous examples, area.py & area2.py? Why might this be?

_____

_____

_____

b. What is the value of `__name__` when we run area3.py as a *script*? _____

c. What might be the value of `__name__` when we *import* area3.py as a module? _____

d. What could would we write to test our hypotheses in questions (b) and (c)?

_____

_____

Note: This is a new, untested POGIL. Let Iris know if there are issues!

Application Questions: Use the Python Interpreter to check your work

1. Write a function that calculates and returns the hypotenuse of a triangle, given the two legs, `a` and `b`. (Note: the math module has a function called `sqrt(x)` that calculates the square root of a value, `x`). Make sure your code can be run as a stand-alone script, but also that it's usable as a module.

Write the *interactive* python commands you'd use to test your function:

>>>

>>>

>>>

>>>

>>>

What would be the output?

Write how you'd run the code as a *script*: _____

What would be the output?

FYI: Typically, when we import a module, immediate output is not desirable. Importing modules is typically limited to reading in *definitions* and *variable assignments*.