Activity 2.1 – Types and Variables

1. Exploring Types

1. Run `python` in a terminal widow. Use the `type` function – *e.g.* `type(1)` – to find the types of the following values. (One of them will produce an error.)

```
1
1.0
"1"
1 + 2
1 + 2.0
1.0 + 2
1.0 + 2.0
"1" + 2
```

2. Typing *e.g.* `type(1.0)` is cumbersome. Based on what you've seen so far, can you tell the type of a value just by looking at the way it's printed?

2. Exploring Variables

1. Without using a computer, what do you expect Python to print during the following session?
2. `>>> x = 10  >>> x  [what goes here?]  >>> y = x  >>> y  [what goes here?]  >>> x = 20  >>> x  [what goes here?]  >>> y  [what goes here?]`
3. Now use Python to test your work. If what you see isn't what you expect, now is an excellent time to talk it over with a classmate or instructor.
4. Draw a *state diagram* of the program in 2. (State diagrams are defined in *Think Python* 2.1. A state diagram is a map of variables to values, drawn as a table of variables and their current values.)

[Optional] Going Beyond

If your mind is at capacity with new concepts, I recommend *not* doing the following exercises today. These are intended for students with prior exposure to programming.

GB1. Turtles all the way down

```
type(1)
type(1.0)
type("1")
type(type(1))
type(type(1.0))
type(type("1"))
type(type(type(1)))
```

Use this to draw a diagram of what values have what types.

GB2. Function types

1. Try the following. (One of these will error.)

```
import math
type(math.sqrt)
type(math.pow)
type(+)
```

2. Add your own function (*e.g.* `is_triangle`) . [This is possible in the Python command-line program, but you may wish to switch to a Jupyter notebook at this point.] Does it have the same type as `math.sqrt`?
3. Does `math.sqrt` takes one argument (`math.sqrt(4)`). `math.pow` takes two arguments (`math.pow(2, 3)`). Do you agree with Python that these are the same type?

GB3. Function values

What do you expect the following to do? Test your hypothesis. How does this compare with another language you know?

```
def add(a, b):
    return a + b

plus = add
print(plus(1, 2))
```

GB4. Closures

See if you can figure out what the following is doing. Is there a similar mechanism in other languages you know?

```
def adder(n):
    def add(a):
        return a + n
    return add

add1 = adder(1)
print(add1(10))

add2 = adder(2)
print(add2(10))
```