Variables, Primitives, and Operators

What we will cover...

1. What is a variable
2. Assigning and overwriting variables
3. Primitive data types (string, int, float, bool, None)
4. Operators (+, -, /, * , %, >, <, ==, !=, is, is not)

Variables

In python, variables are **assigned** with the assignment operator: `=`.

Python naming convention: use **snake_case** for variables (except for specific exceptions, discussed later).

```
my_number = 5
```

Variables

Variables store their value to be used later. We've already seen this use with the `print` function!

```
my_number = 5

print(my_number)
```

Overwriting

Variables in python can be overwritten. Python won't complain.

This can be nice. It can also be dangerous if you don't realize you're overwriting a variable!

To avoid this, we try to minimize the number of variables we have at any time. We will see strategies for this later.

```
my_number = 5

print(my_number)

my_number = 'foo'

print(my_number)
```

Primitive Data Types

Variables can hold all different types of data. Here are some of the main **primitive** data types in Python:

int is an integer (number).
float is a decimal number.
str is a string (characters/words).
bool is a binary value: true or false.
None is a special type for no value!

```
my_int = 5

my_float = 5.0

my_string = 'foo'
my_string = "foo"

my_boolean = True
my_boolean = False

my_none = None
```

Combining values

Often we want to combine two values into a single value.

We can **add** two numbers (ints or floats) with the `+` operator.

We can also **concatenate** two strings with the `+` operator.

Note: in each case, the operator returns the same data type as the two values it was given (more or less).

```
5 + 10

5.0 + 10

'foo' + 'bar'
```

Boolean logic

We can also combine two booleans into a single boolean!

`and` returns true if both values are `True` (or **truthy**).

`or` returns true if either value is `True` (or **truthy**).

```
True and True

True and False

True or False

False or False
```

Math

Numbers have many more arithmetic operators.

What do all the following do?

Once again, each of these operators takes two numbers and returns a single number.

```
5 - 10

5.0 / 10

5 * 10

5 % 2
```

Math

We also might want to compare numbers to each other. We can do that with these comparrison operators.

These operators take two numbers and return what data type?

```
5 < 10

10.0 > 5

5 >= 5

2 <= 5

5 == 5

10 != 5
```

Comparing values

The equality operators (`==` and `!=`) are also used for strings!

The equality operators return a boolean too. This is very useful for checking if a variable holds a certain value.

```
a = 'foo'
b = 'bar'

print(a == b)
print(a != b)
```

Comparing values

The equality operator (`==`) is also used for strings!

The equality operator returns a boolean. This is very useful for checking if a variable holds a certain value.

```
a = 'foo'
b = 'bar'

print(a == b)
```

None

None is a very special value. It represents a lack of value!

It's very important to keep track of missing data. It's also very import to check if a variable is None or not.

We can check if a variable is None with the `is` operator.

```
a = None

a is None
```

Not

The `not` operator returns the opposite of what it is given.

`is` and `not` are also used for **bool** types (`True` and `False`).

```
a = None

a is not None

b = False

b is False
b is not True
```

Differences between == and is

The difference between `==` and `is` is subtle. We will return to this in coming lectures.

For now, remember to use `is` for **bool** and **None** types and `==` for **str**, **int**, and **float** types.

Review

1. What is a variable
2. Assigning and overwriting variables
3. Primitive data types (string, int, float, bool, None)
4. Operators (+, -, /, * , %, >, <, ==, !=, is, is not)