Variables and Data Types: Takeaways

Syntax

- Storing values to variables:

```
twenty = 20

result = 43 + 2**5

currency = 'USD'
```

- Updating the value stored in a variable:

```
x = 30

x += 10 # this is the same as x = x + 10
```

- Rounding a number:

```
round(4.99) # the output will be 5
```

- Using quotation marks to create a string:

```
app_name = "Clash of Clans"

app_rating = '3.5'
```

- Concatenating two or more strings:

```
print('a' + 'b') # prints 'ab'

print('a' + 'b' + 'c') # prints 'abc'
```

- Converting between types of variables:

```
int('4')

str(4)

float('4.3')

str(4.3)
```

- Finding the type of a value:

```
type(4)

type('4')
```

Concepts

- We can store values in the computer memory. Each storage location in the computer's memory is called a **variable**.
- There are two syntax rules we need to be aware of when we're naming variables:
    - We must use only letters, numbers, or underscores (we can't use apostrophes, hyphens, whitespace characters, etc.).
    - Variable names can't start with a number.
- Whenever the syntax is correct, but the computer still returns an error for one reason or another, we say we got a **runtime error**.
- In Python, the `=` operator tells us that the value on the right is **assigned** to the variable on the left. It doesn't tell us anything about equality. We call `=` an **assignment operator**, and we read code like `x = 5` as "five is assigned to x" or "x is assigned five," but not "x equals five."
- In computer programming, values are classified into different **types**, or **data types**. The type of a value offers the computer the required information about how to handle that value. Depending on the type, the computer will know how to store a value in memory, or what operations can and can't be performed on a value.
- In this mission, we learned about three data types: integers, floats, and strings.
- The process of linking two or more strings together is called **concatenation**.

Resources

- More on Strings in Python.