Python Programming: Variables

Lesson Objectives

After this lesson, you will be able to…

- Create and re-assign numerical and string variables.
- Use numerical operators.
- Print complex variable structures.

What’s a Variable?

Turn to the person next to you, and together come up with as many definitions for the word “variable” as you can.

- Consider contexts such as mathematics, the sciences, weather, etc.
- No cheating! Phones off and laptops closed.

Variable

Variables:

- Are boxes that can hold all kinds of information for you.
- Make it easier to store and re-use values.
- Are the most basic piece of code.

To use a variable, we simply announce that we want to use it (we **declare** it).

```
# I've eaten 3 cupcakes
cupcakes_ive_eaten = 3
print(cupcakes_ive_eaten)
# Prints 3
```

Naming Conventions: Mistakes and Syntax

Some common naming mistakes:

- Not using meaningful names. `delicious = 3` doesn't mean anything - `cupcakes_ive_eaten = 3` does!
- Case sensitivity (`CUPCAKES_IVE_EATEN` and `cupcakes_ive_eaten` are not the same!)
- No spaces or punctuation ("cupcakes i've eaten" isn't allowed)
  - This is invalid **syntax**
  - Use snake_case: `lowercase_letters_with_underscores` (it's in the official Python style guide)

Discussion: Changing Values

What if, later, you eat more cupcakes? Now, this is wrong.

```
cupcakes_ive_eaten = 3
```

What do you think we need to do?

Discussion: Reassigning Variables

In the example below, what do you think the output of the code is?

```
cupcakes_ive_eaten = 3

print(cupcakes_ive_eaten)

cupcakes_ive_eaten = 4

print(cupcakes_ive_eaten)
```

Quick Review

- A variable is a box that holds a value.
- It can be declared, called, and changed within your program.
- When declaring variables, syntax and naming conventions matter!
- Variables can be reassigned as often as you like, but only the most recent declaration counts.

UP NEXT: Math!

Mathematical Operators

Math works on numerical variables, too!

- The `+`, `-`, `*` (multiply), and `/` (divide) operators work just like they do with regular math.

```
cupcakes_ive_eaten = 6 + 3
print(cupcakes_ive_eaten)
# Prints 9


cupcakes_ive_eaten = 6 - 3
print(cupcakes_ive_eaten)
# Prints 3


cupcakes_ive_eaten = 6 * 3
print(cupcakes_ive_eaten)
# Prints 18
```

Even More Mathematical Operators

Beyond the `+`, `-`, `*` (multiply), and `/` (divide) operators, we have modulus and exponents.

```
making_exponents = 10 ** 2

print(making_exponents)

# Prints 100


more_exponents = 10 ** 3

print(more_exponents)

# Prints 1,000


making_modulus = 10 % 3

print(making_modulus)

# Prints 1
```

Math On The Same Variable

You can reassign a variable *using that very same variable* - or other variables!

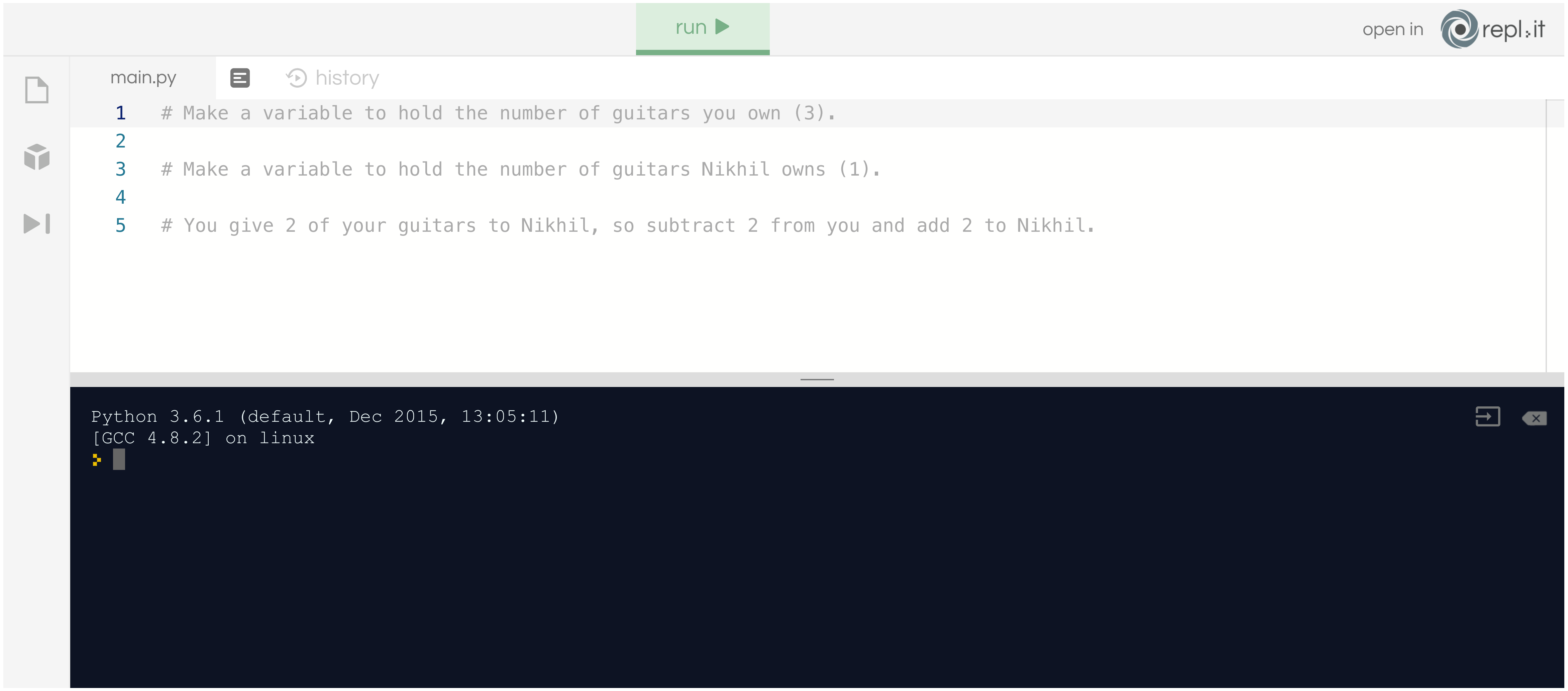
```
cupcakes_ive_eaten = 3

cupcakes_ive_eaten = cupcakes_ive_eaten + 1

print(cupcakes_ive_eaten)

# Prints 4.

cupcakes_left_in_box = 6

cupcakes_left_in_box = cupcakes_left_in_box - 1 print(cupcakes_ive_eaten)

# Prints 5.

cupcakes_left_in_box = cupcakes_left_in_box - cupcakes_ive_eaten print(cupca

# Prints 1.
```

Partner Exercise: Mathematical Operators

Pair up and choose roles:

- Driver
- Navigator

Try to code the below:



```
# Make a variable to hold the number of guitars you own (3).

# Make a variable to hold the number of guitars Nikhil owns (1).

# You give 2 of your guitars to Nikhil, so subtract 2 from you and add 2 to Nikhil.
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

Reassignment Shorthand

This is okay:

```
my_num = 9
my_num = my_num + 7
# my_num is now 16
```

But this is better:

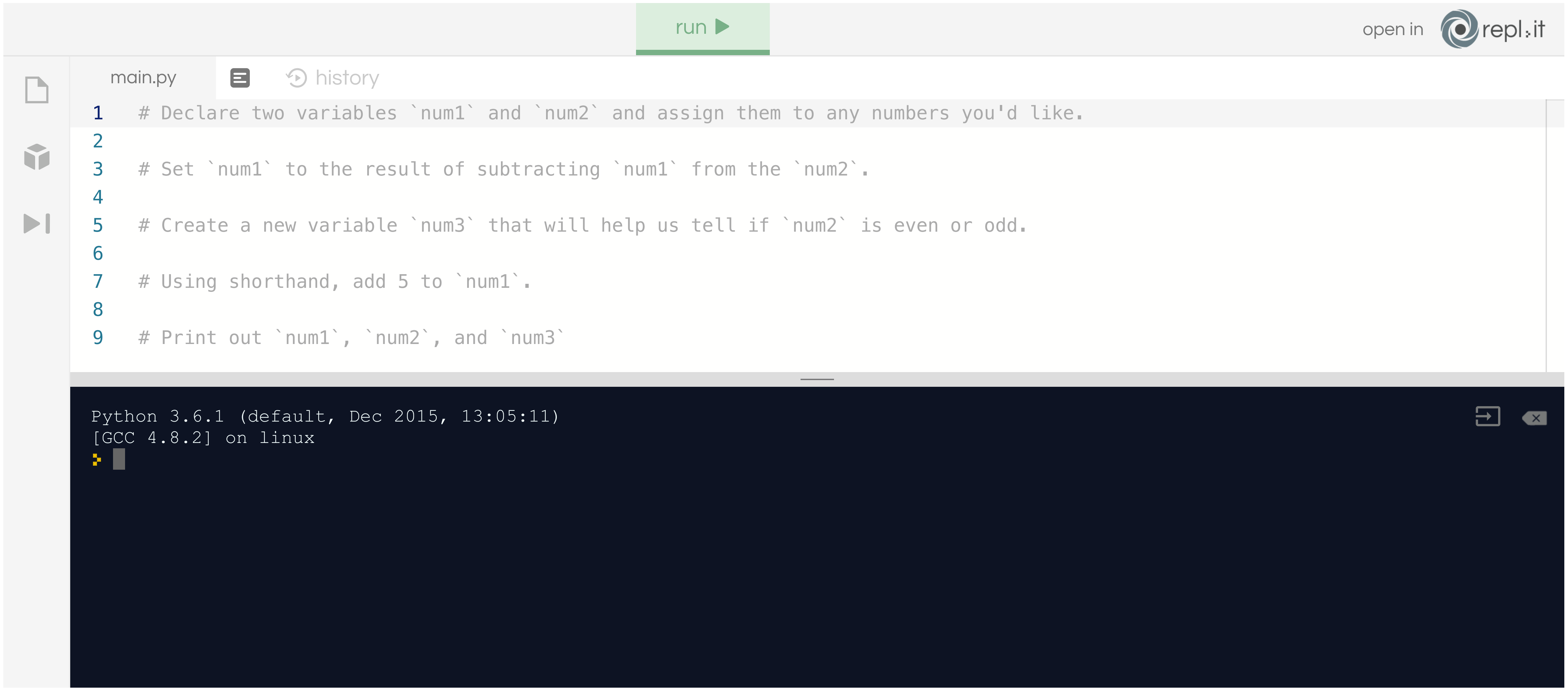
```
my_num = 9
my_num += 7 # += is short for theSameVariable = theSameVariable + 7
# my_num is now 16
```

This works with `+=`, `-=`, `*=`, `/=` - any math operations.

Partner Exercise: Numerical Reassignment

Get with the same partner, but switch driver and navigator roles.

In the environment below, follow the prompts:



```python
# Declare two variables `num1` and `num2` and assign them to any numbers you'd like.

# Set `num1` to the result of subtracting `num1` from the `num2`.

# Create a new variable `num3` that will help us tell if `num2` is even or odd.

# Using shorthand, add 5 to `num1`.

# Print out `num1`, `num2`, and `num3`
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
```

Important Aside: Even or Odd?

Is 6 even or odd?

Is 7 even or odd?

How do you think a computer knows?

Modulus operator shows the remainder of a division problem.

Modding by 2 only gives a `0` or a `1`.

- **4 % 2**:
  - `4 % 2 = 0`. Even!
- **5 % 2**:
  - `5 % 2 = 1`. Odd!

Quick Review

- A variable is a value that can be defined, declared, called and changed within your program.
  - `my_number = 5`
- Naming:
  - Variable names are case sensitive.
  - Use `snake_case`!
- Variables can be reassigned as often as you like, but only the most recent declaration counts.
- Python can do math using operators, such as `+`, `-`, `*`, and `/`
  - You can shorthand the math assignments: `my_num += 7`

Taking a Breather

That was a lot of math!

When it comes down to it, computers operate with a simple, straightforward logic.

Let’s switch gears. Up next: Strings!

Introducing Strings

A *character* is:

- Anything on your keyboard , such as a letter or a number.
- “Apple” is five characters: a, p, p, l, e.
- Spaces count! (they’re on the keyboard!)

A *string* is:

- A complete list of characters.
- “Apple”
- “Chocolate Cupcake”
- This entire sentence: “Hello, you are 1 of a kind!”

How Do I Create Strings in Python?

You tell Python that your variable will hold a string using quotation marks.

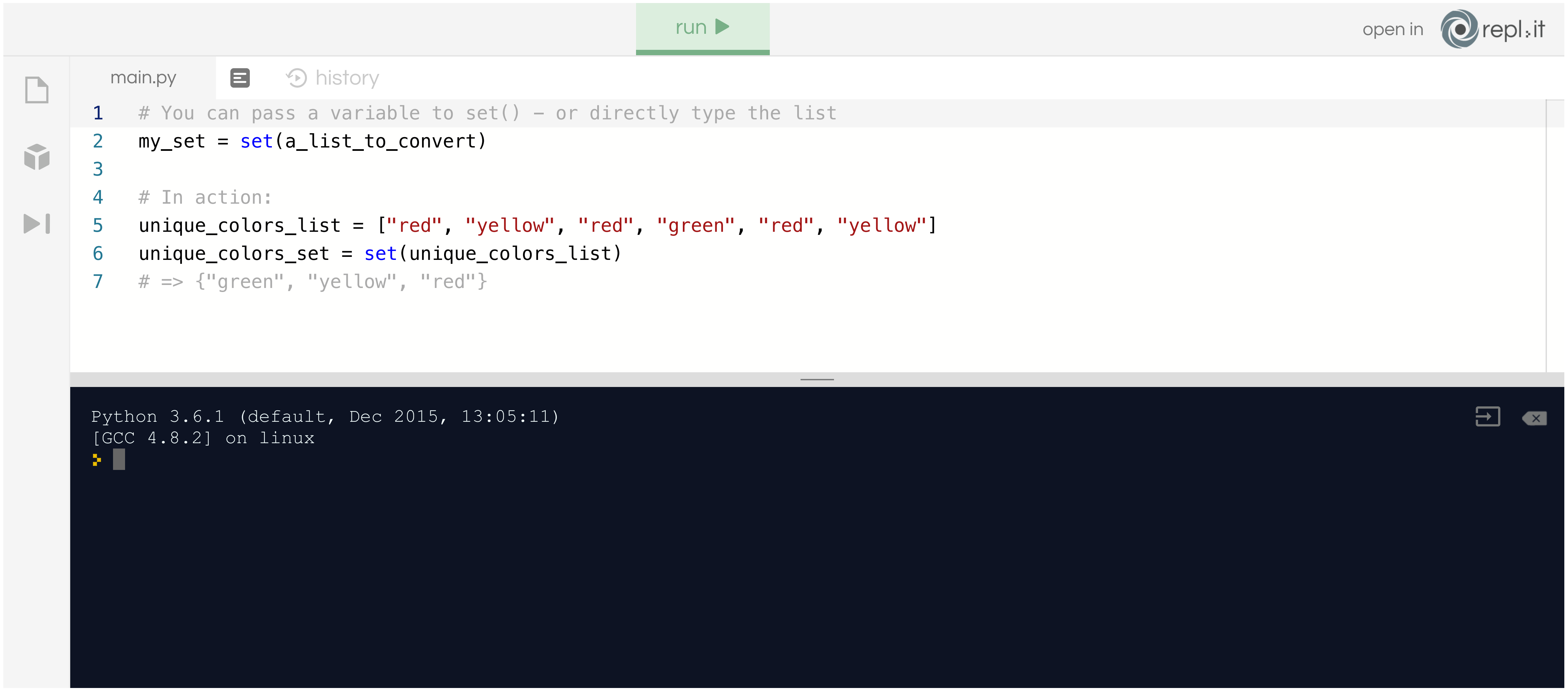
```
box_contents = "cupcakes" # This is a string
print(box_contents) # It's a normal variable - we can print it.
best_snack = "Frosted Cupcakes" # This is a string.
cupcakes_ive_eaten = 5 # No quotes - this is a number.
cupcakes_ive_eaten_as_string = "5" # Because this is in quotes, this is a st
```

We Do: Declaring Strings

A “We Do” means let’s practice together. Follow along!

1. We’ll declare a variable called `name` and assign it the value `Marty`
2. We’ll declare a variable called `car` and assign it the value `Delorean`
3. We’ll declare a variable called `speed` and assign it the *string* value `"88"`
4. We’ll print out these variables
5. We’ll add `4` to `speed`- what happens?

We Do: Declaring Strings



```python
# You can pass a variable to set() - or directly type the list
my_set = set(a_list_to_convert)

# In action:
unique_colors_list = ["red", "yellow", "red", "green", "red", "yellow"]
unique_colors_set = set(unique_colors_list)
# => {"green", "yellow", "red"}
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
›
```

String Concatenation

`+` on:

- Numerical variables adds (`5 + 5 = 10`).
- String variables *concatenate* (`"Doc" + "Brown" = "DocBrown"`).
  - *Pssst: Pronunciation tip: con-CAT-en-ATE*
- Numerical strings concatenate to new strings! (`"5" +`“4”`=`“54”`)

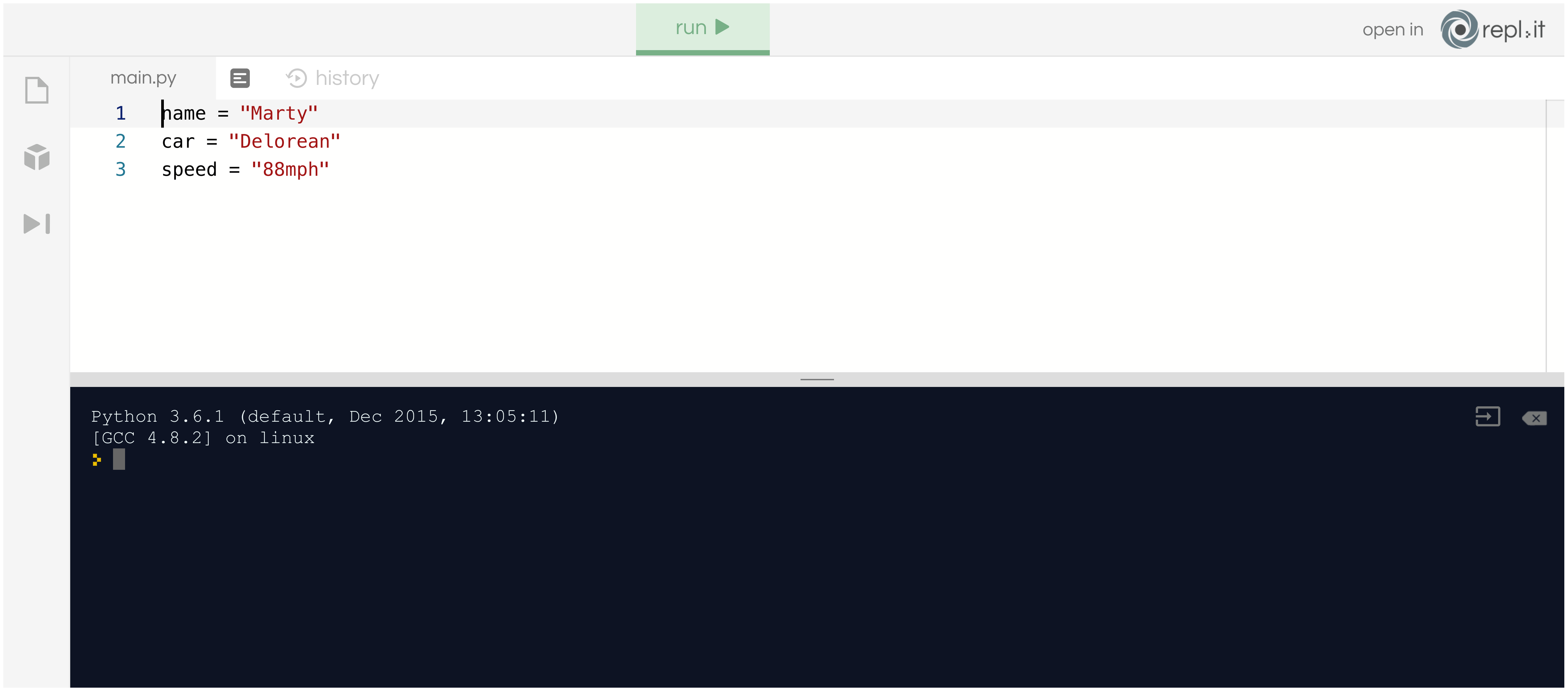
```
first_name = "Doc"
last_name = "Brown"
full_name = first_name + last_name
print full_name
# Prints "DocBrown".
```

We Do: Spaces in Concatenation

It's another "We Do." Let's do this together - follow along!

To begin: `sentence = name + "is driving his" + car + speed`

We expect the sentence to be `Marty is driving his Delorean 88mph`. Is that what we got?



```python
name = "Marty"
car = "Delorean"
speed = "88mph"
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> 
```

Strings and Printing: Review

Strings are made with quotes:

```
name = "Marty"
car = "Delorean"
speed = "88"
```

String Concatenation - we need to add the spaces!

```
sentence = name + " is driving his " + car + " " + speed
string_numbers = "88" + "51"
# string_numbers = 8851
```

To easily create spaces while printing:

```
print(name, "is driving his", car, speed)
```

Discussion: Some Common Mistakes: 1

Do you think this will run? If yes, what does it print?

```
my_num

print(my_num)
```

Discussion: Some Common Mistakes: 2

How about this? Does it run? If so, what does it print?

```
my_num = 5

print()
```

Discussion: Some Common Mistakes: 3

How about this? Does it run? If so, what does it print?

```
my_num = 5

my_string = "Hello"

print(my_num + my_string)
```

Discussion: Some Common Mistakes: 4

One last question. What does this do?

```
my_num1 = "10"
my_num2 = "20"
print(my_num1 + my_num2)
```

Q&A and Summary

We learned a lot today!

- We created, used, and re-assigned number and string variables.
- We used the numerical operators `+ - / * // %`
- We did some complex stuff with the `print` function!

Congrats! You’ve finished your first programming lesson!

Additional Resources

- A Repl.it Summarizing Print Statements
- Python For Beginners
- Python Programming Tutorial: Variables
- Variables in Python
- Operators Cheatsheet
- Python Style Guide: Naming
- Python-Strings
- String Concatenation and Formatting
- String Concatenation and Formatting - Video