Expressions

Chris Piech and Mehran Sahami
CS106A, Stanford University

Recall, add2numbers.py Program

```
def main():
    print("This program adds two numbers.")
    num1 = input("Enter first number: ")
    num1 = int(num1)
    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

Recall, add2numbers.py Program

```
def main():
    print("This program adds two numbers.")
    num1 = int(input("Enter first number: "))

    num2 = input("Enter second number: ")
    num2 = int(num2)
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

Recall, add2numbers.py Program

```
def main():
    print("This program adds two numbers.")
    num1 = int(input("Enter first number: "))

    num2 = int(input("Enter second number: "))

    total = num1 + num2
    print("The total is " + str(total) + ".")
```

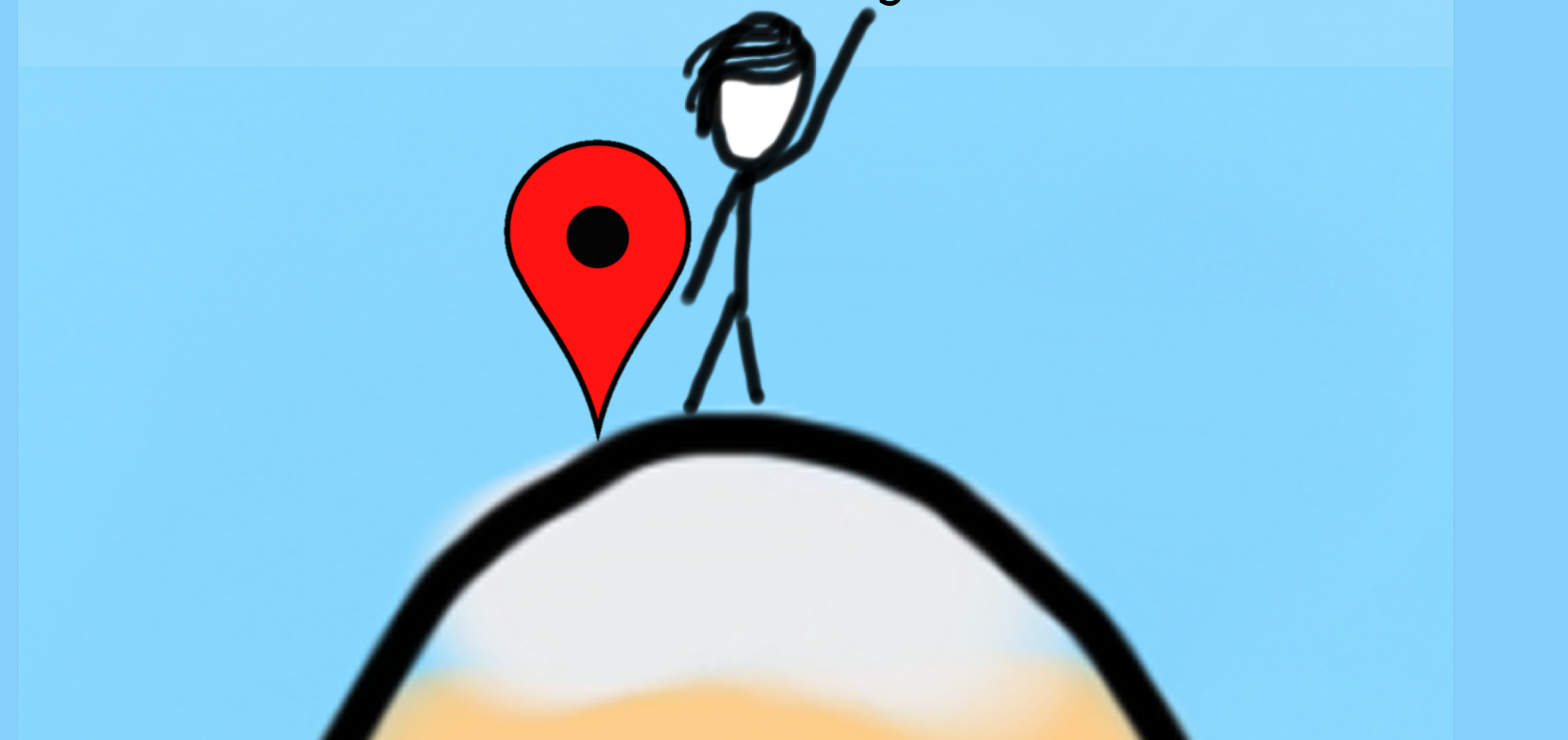
Recall, add2numbers.py Program

```
def main():
    print("This program adds two numbers.")
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    total = num1 + num2
    print("The total is " + str(total) + ".")
```

- Often, this is how you'll see code that gets input
- But, what if I want to do more than add?
- It's time for the world of *expressions*

Today's Goal

1. Understanding arithmetic expressions
2. Using constants
3. Random number generation



Arithmetic Operators

```
num1 = 5
num2 = 2
```

- Operations on numerical types (**`int`** and **`float`**)
- Operators

			num3
`+`	"addition"	Ex.: `num3 = num1 + num2`	7
`-`	"subtraction"	Ex.: `num3 = num1 - num2`	3
`*`	"multiplication"	Ex.: `num3 = num1 * num2`	10
`/`	"division"	Ex.: `num3 = num1 / num2`	2.5
`//`	"integer division"	Ex.: `num3 = num1 // num2`	2
`%`	"remainder"	Ex.: `num3 = num1 % num2`	1
`**`	"exponentiation"	Ex.: `num3 = num1 ** num2`	25
`-`	"negation" (unary)	Ex.: `num3 = -num1`	-5

Precedence

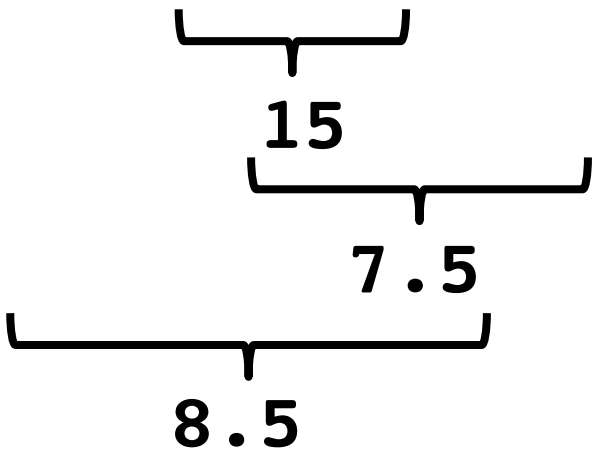
- Precedence of operator (in order)

`()`	"parentheses"	highest
`**`	"exponentiation"	
`-`	"negation" (unary)	
`*, /, //, %`		
`+, -`		lowest

- Operators in same precedence category are evaluated left to right
  - Similar to rules of evaluating expressions in algebra

Precedence Example

```
x = 1 + 3 * 5 / 2
```



x | 8.5

Implicit Type Conversion

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Operations on two **int**s (except **/**) that would result in an integer value are of type **int**

  num1 + 7 = 12 (int)

  – Dividing (**/**) two **int**s results in a **float**, even if result is a round number (Ex.: **6 / 2** = **3.0**)

- If either (or both) of operands are **float**, the result is a **float**

  num3 + 1 = 2.9 (float)

- Exponentiation depends on the result:

  num2 ** 3 = 8 (int)
  2 ** -1 = 0.5 (float)

Explicit Type Conversion

```
num1 = 5
num2 = 2
num3 = 1.9
```

- Use **`float(`*value*`)`** to create new real-valued number

  `float(num1)` = **`5.0`** **`(float)`**

  – Note that **`num1`** is not changed. We created a new value.

  `num1 + float(num2)` = **`7.0`** **`(float)`**

  `num1 + num2` = **`7`** **`(int)`**

- Use **`int(`*value*`)`** to create a new integer-valued number (truncating anything after decimal)

  `int(num3)` = **`1`** **`(int)`**

  `int(-2.7)` = **`-2`** **`(int)`**

Float is Not Always Exact

```
num1 = 5
num2 = 2
num3 = 1.9
```

- What is type of: **`num3 - 1`**
  - Answer: **`float`**
- What is value of: **`num3 - 1`**
  - Answer: **`0.8999999999999999`**
  - WHAT?!

Expression Shorthands

```
num1 = 5
num2 = 2
num3 = 1.9
```

num1 = num1 + 1	same as	**num1 += 1**
num2 = num2 - 4	same as	**num2 -= 4**
num3 = num3 * 2	same as	**num3 *= 2**
num1 = num1 / 2	same as	**num1 /= 2**

- Generally:

  variable **=** ***variable*** operator (***expression***)

  is same as:

  variable operator**=** ***expression***

Let's consider an example

average2numbers.py

average2numbers.py

```
"""
File: average2numbers.py
------------------------
This program asks the user for two numbers
and prints their average.
"""

def main():
    print("This program averages two numbers.")
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    total = (num1 + num2) / 2
    print("The average is " + str(total) + ".")

# This provided line is required at the end of a
# Python file to call the main() function.
if __name__ == '__main__':
    main()
```


Constants

```
INCHES_IN_FOOT = 12
PI = 3.1415
```

- Constants make code easier to read (good style):

```
area = PI * (radius ** 2)
```

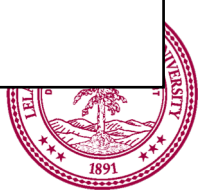
  - Written in all capital SNAKE_CASE with descriptive names
  - Constant are really variables that represent quantities that don't change while the program is running
  - Can be changed between runs (as necessary)
    - "Hey, we need to compute a trajectory to get us to Mars"

```
PI = 3.141592653589793
```

  - Code should be written with constants in a general way so that it still works when constants are changed

Example of Using Constants

```
"""
File: constants.py
------------------
An example program with constants
"""

INCHES_IN_FOOT = 12

def main():
    feet = float(input("Enter number of feet: "))
    inches = feet * INCHES_IN_FOOT
    print("That is " + str(inches) + " inches!")

# This provided line is required at the end of a Python file
# to call the main() function.
if __name__ == '__main__':
    main()
```

Python `math` Library

```
import math
```

- math library has many built-in constants:

`math.pi`	mathematical constant π
`math.e`	mathematical constant e

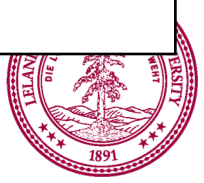
- and useful functions:

`math.sqrt(`x`)`	returns square root of x
`math.exp(`x`)`	returns e^x
`math.log(`x`)`	returns natural log (base e) of x

- These are just a few examples of what's in math

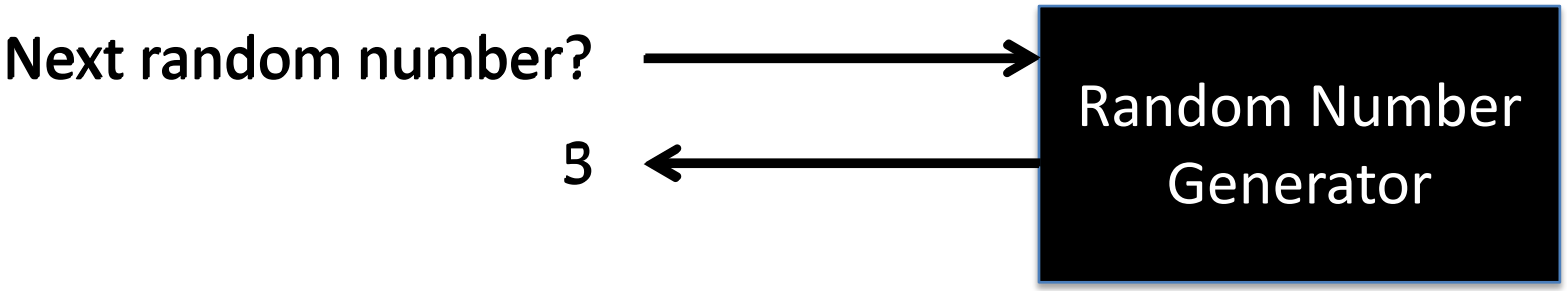
Example of Using `math` Library

```
"""
File: squareroot.py
-------------------
This program computes square roots
"""

import math

def main():
    num = float(input("Enter number: "))
    root = math.sqrt(num)
    print("Square root of " + str(num) + " is " + str(root))

# This provided line is required at the end of a Python file
# to call the main() function.
if __name__ == '__main__':
    main()
```



Random Number Generation

- Want a way to generate random number
  - Say, for games or other applications
- No "true" randomness in computer, so we have *pseudorandom* numbers
  - "That looks pretty random to me"
- Want "black box" that we can ask for random numbers



- Can "seed" the random number generator to always produce the same sequence of "random" numbers

Python `random` Library

```
import random
```

Function	What it does
`random.randint(`***min, max***`)`	Returns a random integer between *min* and *max*, inclusive.
`random.random()`	Returns a random real number (float) between 0 and 1.
`random.uniform(`***min, max***`)`	Returns a random real number (float) between *min* and *max*.
random.seed(***x***)	Sets "seed" of random number generator to *x*.

Let's consider an example

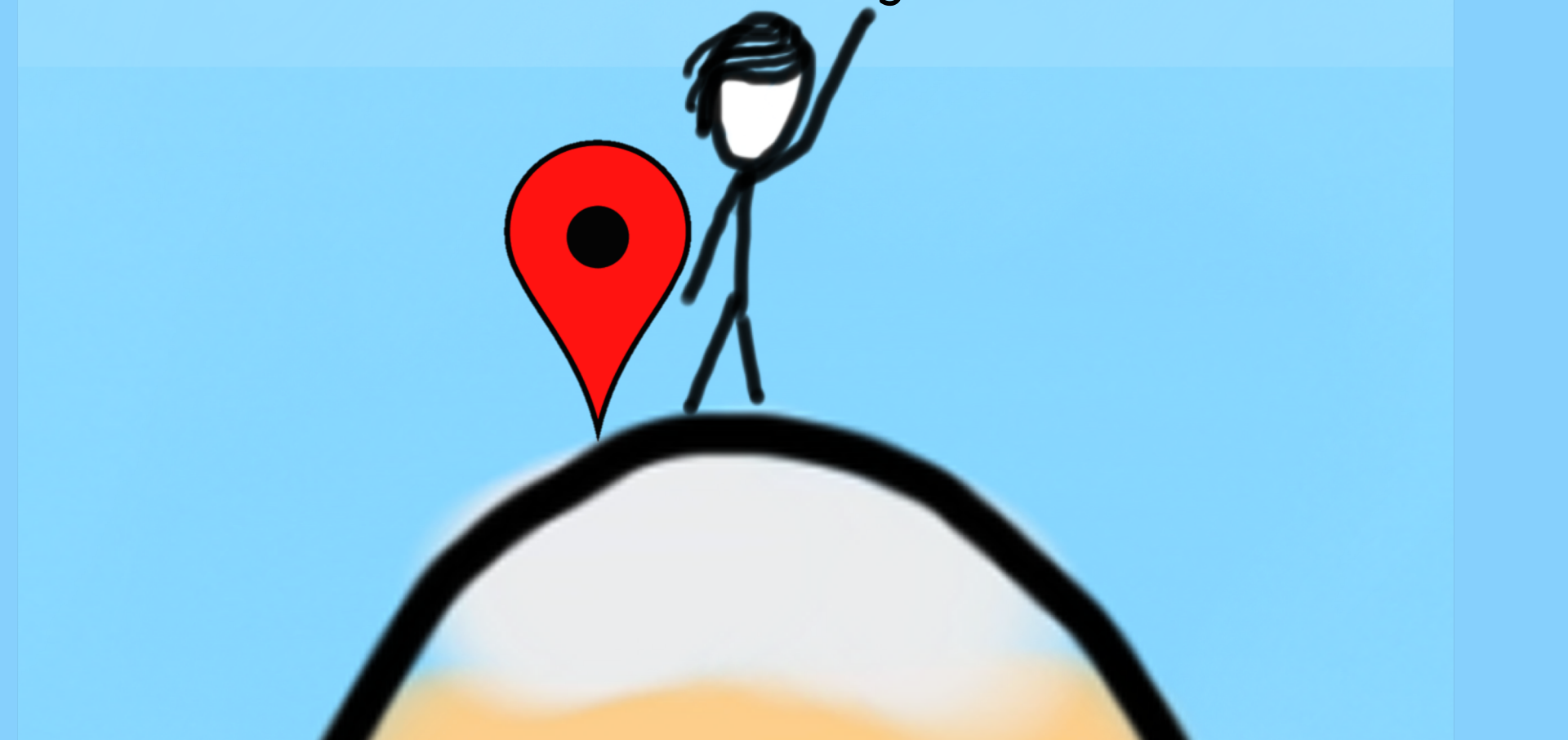
rolldice.py

Example of Using `random` Library

```
"""
File: rolldice.py
-----------------
Simulate rolling two dice
"""

import random

NUM_SIDES = 6

def main():
    # setting seed is useful for debugging
    # random.seed(1)
    die1 = random.randint(1, NUM_SIDES)
    die2 = random.randint(1, NUM_SIDES)
    total = die1 + die2
    print("Dice have " + str(NUM_SIDES) + " sides each.")
    print("First die: " + str(die1))
    print("Second die: " + str(die2))
    print("Total of two dice: " + str(total))
```

Today's Goal

1. Understanding arithmetic expressions
2. Using constants
3. Random number generation



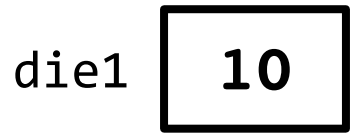
Putting it all together: dicesimulator.py

What's Going On?

```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```

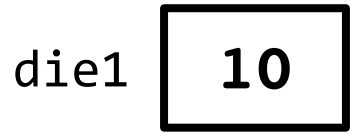
What's Going On?

```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```



What's Going On?

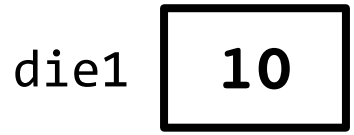
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
```

What's Going On?

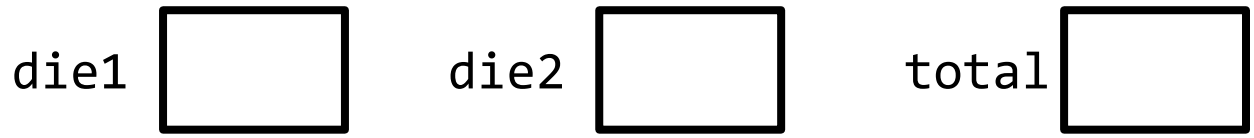
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
```

What's Going On?

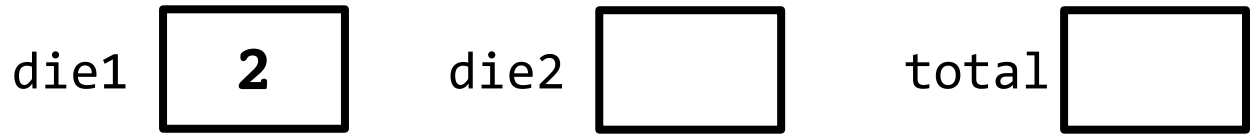
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```



die1 in main() starts as: 10

What's Going On?

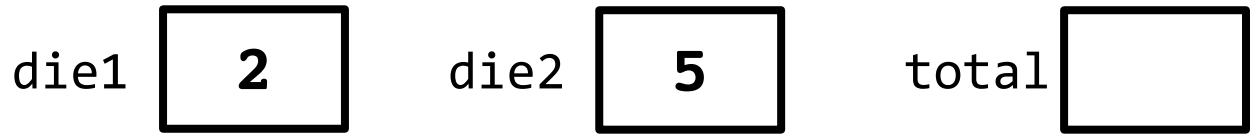
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```



die1 in main() starts as: 10

What's Going On?

```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```



die1 in main() starts as: 10

What's Going On?

```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```

die1	2	die2	5	total	7

die1 in main() starts as: 10

What's Going On?

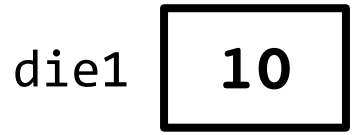
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```

die1	2	die2	5	total	7

```
die1 in main() starts as: 10
Total of two dice: 7
```


What's Going On?

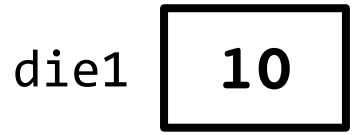
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

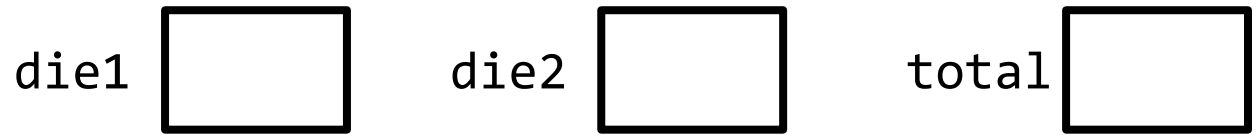
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

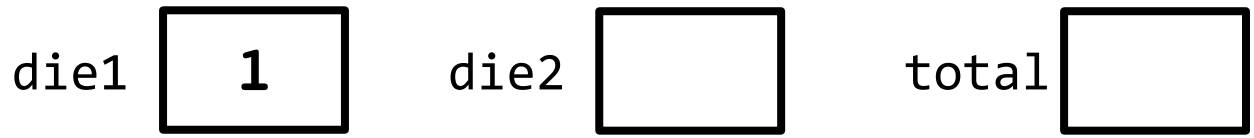
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```




```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

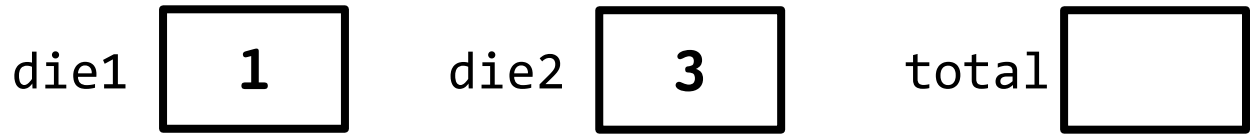
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```




```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```




```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```

die1	1	die2	3	total	4

```
die1 in main() starts as: 10
Total of two dice: 7
```

What's Going On?

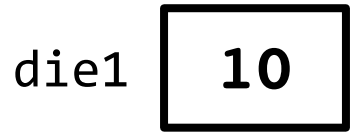
```
def main():
    def roll_dice():
        die1 = random.randint(1, NUM_SIDES)
        die2 = random.randint(1, NUM_SIDES)
        total = die1 + die2
        print("Total of two dice: " + str(total))
```

die1	1	die2	3	total	4

```
die1 in main() starts as: 10
Total of two dice: 7
Total of two dice: 4
```


What's Going On?

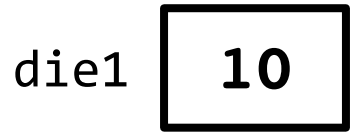
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
Total of two dice: 4
```

What's Going On?

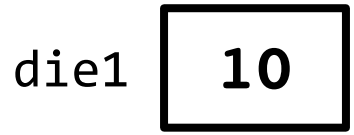
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
Total of two dice: 4
```

What's Going On?

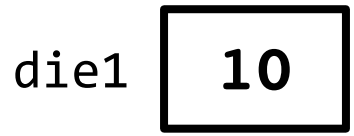
```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
Total of two dice: 4
Total of two dice: 5
```

What's Going On?

```
def main():
    die1 = 10
    print("die1 in main() starts as: " + str(die1))
    roll_dice()
    roll_dice()
    roll_dice()
    print("die1 in main() is: " + str(die1))
```




```
die1 in main() starts as: 10
Total of two dice: 7
Total of two dice: 4
Total of two dice: 5
die1 in main() is: 10
```

You're rockin' it!