CPSC 231 Tutorial #6

michael-hung.ca/teaching

Reminders

TOMORROW

Assignment 2 Individual Component Due

NEXT TUESDAY

Quiz #4

Truth Tables

`(A and B) or (not A or B)`

A	not A	B	not A or B	A and B	(A and B) or (not A or B)
T	F	T	T	T	?
T	F	F	F	F	?
F	T	T	T	F	?
F	T	F	T	F	?

Truth Tables

`(A and B) or (not A or B)`

A	not A	B	not A or B	A and B	(A and B) or (not A or B)
T	F	T	T	T	T
T	F	F	F	F	F
F	T	T	T	F	T
F	T	F	F	F	F

Getting Input

- From the command line

```
import sys
...
arg1 = sys.argv[1]
arg2 = sys.argv[2]
...
```

- During runtime

```
myInput = input(“Enter something...");
```

Parsing a String to a Number

Input from command line and input() both return strings

For some string **s**

- int(s) *turns **s** into an integer*
- float(s) *turns **s** into a float*

Python will throw a **ValueError** if the types don’t match up

- e.g. int(“one”)

Turning a Number into a String

For some number **n**

```
    str(n)
```

Operations

Addition	+	1 + 1 = 2
Subtraction	-	2 - 1 = 1
Multiplication	*	2 * 2 = 4
Division	/	3 / 2 = 1.5
Exponents	**	3 ** 2 = 9
Integer Division	//	3 // 2 = 1
Modulus	%	11 % 5 = 1

Operations

Add to Self	+=
Subtract from Self	-=
Multiply Self	*=
Divide By Self	/=

Handling Errors

```
try:

    x = int(input(“Enter an integer: ”))
    break

except (ValueError):

    print(“Not a valid integer”)


...
```

The “random” module

```
import random
start = 1
stop = 11

number = random.randrange(start, stop)

# gets a random integer between 1 and 10, inclusive.
```

Recap

- Truth Tables
- Getting Input
  - Command Line
  - During Runtime
- Parsing a String to a Number
- Turning a Number to a String
- Integer and Float Operations
- Self-Operations
- Error Handling
- ‘random’ module

Writing to Terminal

- `print()`
  - Takes a string argument
  - Can take multiple arguments separated by commas
    - Will automatically separate each with a space character (“ ”)
  - Will automatically end with a newline character (“\n”)
- `stdio.write()`
  - Takes **only** one string argument
  - Does **not** end with a newline character
- `stdio.writeln()`
  - Takes **only** one string argument
  - Will automatically end with a newline character (“\n”)

String Things

- Concatenation (str_1 + str_2 + … str_n)
  - “1” + “1” + “1” → “111”
  - “1” + 1 → Error
- Special Characters
  - “\t” → Tab Character
  - “\n” → Newline Character
- Quotation marks
  - “A string”
  - ‘Another string’
  - “I’m a string, too”
  - ‘“Me too!”, I said’
  - ““““He replied, “I’m not so sure,” slightly puzzled.””””

Some Exercises!

Order Check

INPUT

Three floats: **x**, **y**, and **z**

OUTPUT

True	if values are strictly ascending in order (e.g. x < y < z, x > y > z)
False	otherwise

Converting RGB to CMYK

INPUT
Three integers **r**, **g**, and **b**. Assume each is between 0 - 255, inclusive.

OUTPUT
The values of **c**, **m**, **y**, and **k**.

Hints:
If RGB values are all 0, then CMY values are also 0 and K is 1.

```
w = max (r / 255, g / 255, b / 255)
c = (w - (r / 255)) / w
m = (w - (g / 255)) / w
y = (w - (b / 255)) / w
k = 1 - w
```

FizzBuzz

INPUT

An integer **n**

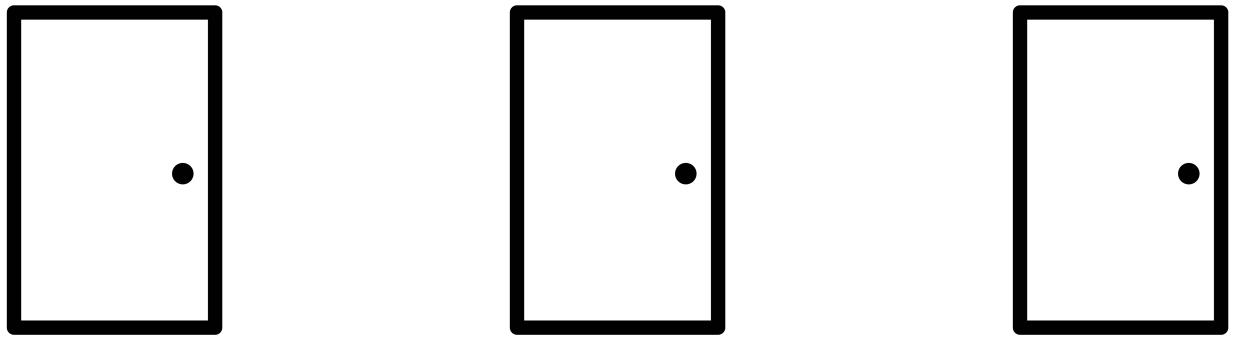
OUTPUT

Print all numbers from 1 to n, but:

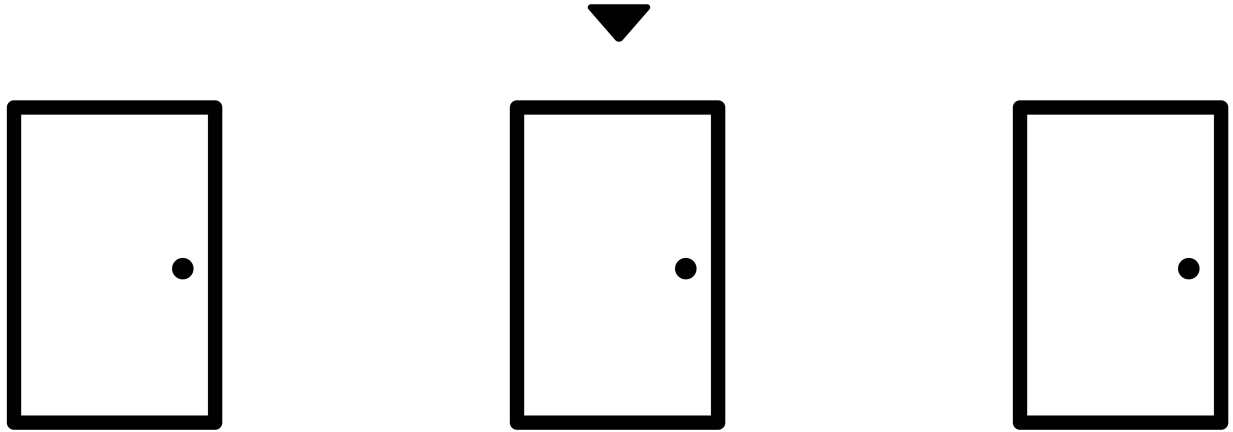
- For multiples of 3, print “Fizz” instead
- For multiples of 5, print “Buzz” instead
- For multiples of both, print “FizzBuzz” instead

CHALLENGE: DO NOT USE NESTED IF BLOCKS.

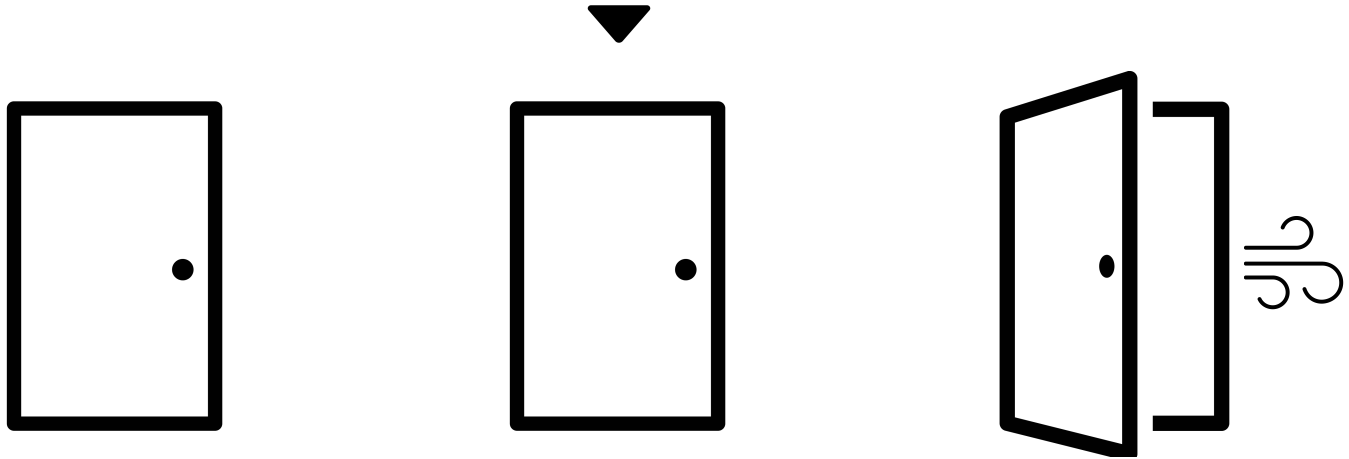
The Monty Hall Problem



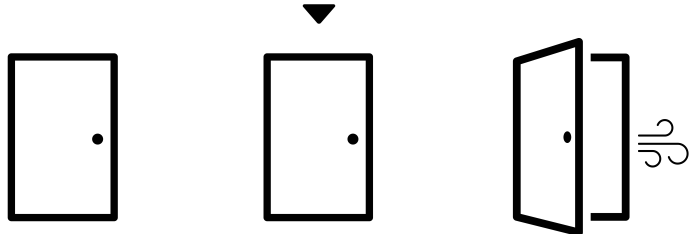
The Monty Hall Problem



The Monty Hall Problem



The Monty Hall Problem



INPUT

An integer **n**, the number of times to play the game

OUTPUT

Print the **win percentage** while using **Keep-Same** strategy
Print the **win percentage** while using **Switch** strategy

The Monty Hall Problem

Choose Door 1

Door 1	Door 2	Door 3	Keep Same	Switch
Prize	Nothing	Nothing	Win	Lose
Nothing	Prize	Nothing	Lose	Win
Nothing	Nothing	Prize	Lose	Win

Debugging in VS Code