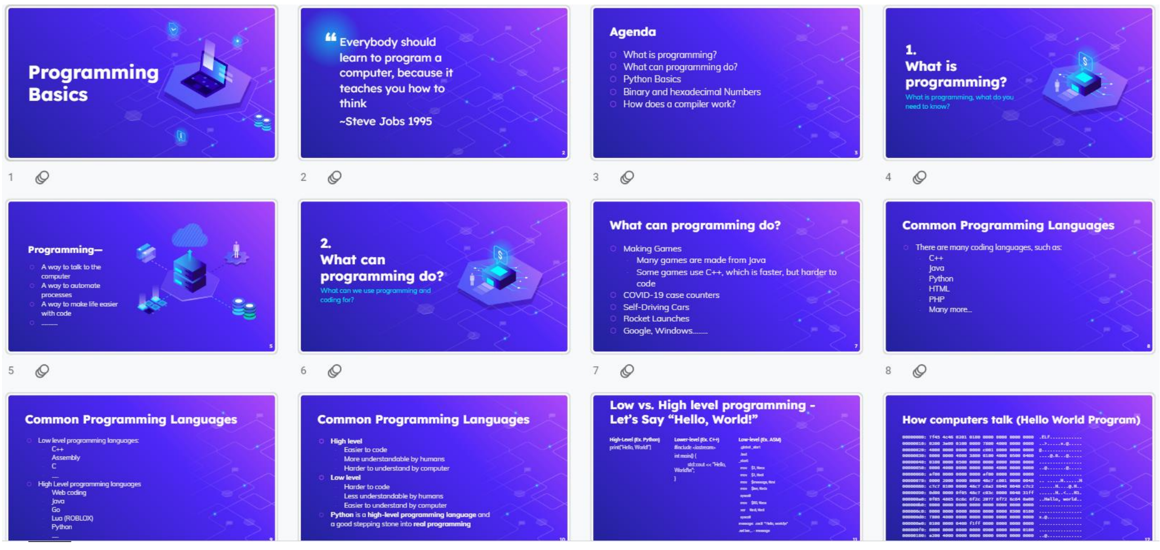
Felix's Coding Lesson Plans

Grade 8 Gifted Class Coding Lesson Plan

When I was preparing for this lesson plan, I reflected on the time I did a short presentation about coding basics when I was in Grade 6 and believed that most of my classmates at that time didn't really understand what I was presenting. I need to do it better this time and really help my friends in school learn real coding! I reviewed some online learning materials available through Waterloo's open courseware website to help with some ideas on how to teach an introduction to coding lesson without talking about specific coding languages.
I also borrowed some concepts and ideas, as well as student feedback from the coding lessons I taught for Coding-is-Fun targeting kids with and without coding experiences.
I took into consideration that my schoolmates from the gifted program all had experience learning block coding (Scratch) in Grade 6.

- Objectives
    - Understand the math behind programing
    - Able to use Python to program simple games
- Lessons
    - Understand programing basics and introduction to Python (1x1.5hrs lesson)
        - What is programming?
        - What can programming do?
        - Python Basics – "Hello, World!" on repl.it.
        - Binary and hexadecimal Numbers
        - How does a compiler work?
    - Number Guessing Game (2x1.5hrs lessons, 2nd lesson is a Q&A and working session)
        - Flow of game
        - If statements
        - Logic orders
        - Random Numbers
        - String operations
        - Loops
    - Text-Based Adventure Game (2x1.5hrs lessons, 2nd lesson is a Q&A and working session)
        - What are text-based adventure games
        - Examples
        - Functions
        - In-place Operators
        - Coloured text
        - Homework
    - Showcase of the games developed (1x1.5hrs lesson)

- Links to the Slide Decks used:



1

Programming Basics

2

"Everybody should learn to program a computer, because it teaches you how to think

~Steve Jobs 1995

3

Agenda

- What is programming?
- What can programming do?
- Python Basics
- Binary and hexadecimal Numbers
- How does a compiler work?

4

1. What is programming?

What is programming, what do you need to know?

5

Programming—

- A way to talk to the computer
- A way to automate processes
- A way to make life easier with code
- ………

6

2. What can programming do?

What can we use programming and coding for?

7

What can programming do?

- Making Games
  - Many games are made from Java
  - Some games use C++, which is faster, but harder to code
- COVID-19 case counters
- Self-Driving Cars
- Rocket Launches
- Google, Windows……..

8

Common Programming Languages

- There are many coding languages, such as:
  - C++
  - Java
  - Python
  - HTML
  - PHP
  - Many more…

9

Common Programming Languages

- Low level programming languages:
  - C++
  - Assembly
  - C
  - ….
- High Level programming languages
  - Web coding
  - Java
  - Go
  - Lua (ROBLOX)
  - Python
  - ….

10

Common Programming Languages

- **High level**
  - Easier to code
  - More understandable by humans
  - Harder to understand by computer
- **Low level**
  - Harder to code
  - Less understandable by humans
  - Easier to understand by computer
- **Python** is a **high-level programming language** and a good stepping stone into **real programming**

11

Low vs. High level programming - Let's Say "Hello, World!"

High-Level (Ex. Python)

```
print("Hello, World!")
```

Lower-level (Ex. C++)

```
#include <iostream>
int main() {
        std::cout << "Hello, World!\n";
}
```

Low-level (Ex. ASM)

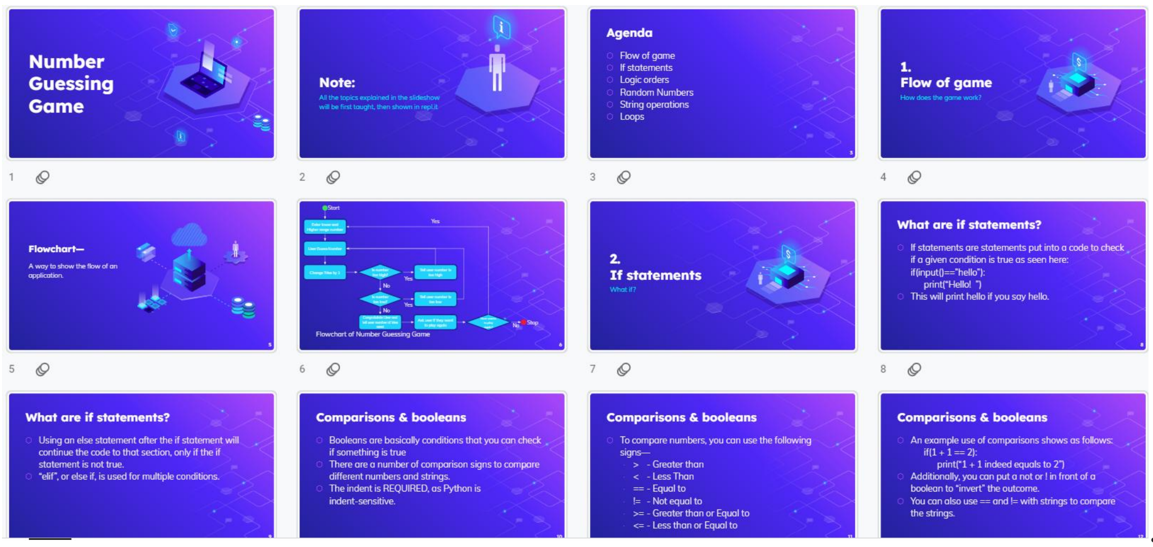
```
 .global _start
 .text
_start:
 mov   $1, %rax
 mov   $1, %rdi
 mov   $message, %rsi
 mov   $len, %rdx
 syscall
 mov   $60, %rax
 xor   %rdi, %rdi
 syscall
message: .ascii "Hello, world\n"
.set len, . - message
```

12

How computers talk (Hello World Program)

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0200 3e00 0100 0000 7800 4000 0000 0000  ..>.....x.@.....
00000020: 4000 0000 0000 0000 c001 0000 0000 0000  @...............
00000030: 0000 0000 4000 3800 0100 4000 0500 0400  ....@.8...@.....
00000040: 0100 0000 0500 0000 0000 0000 0000 0000  ................
00000050: 0000 4000 0000 0000 0000 4000 0000 0000  ..@.......@.....
00000060: af00 0000 0000 0000 af00 0000 0000 0000  ................
00000070: 0000 2000 0000 0000 48c7 c001 0000 0048  .. .....H......H
00000080: c7c7 0100 0000 48c7 c6a2 0040 0048 c7c2  ......H....@.H..
00000090: 0d00 0000 0f05 48c7 c03c 0000 0048 31ff  ......H..<...H1.
000000a0: 0f05 4865 6c6c 6f2c 2077 6f72 6c64 0a00  ..Hello, world..
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0300 0100  ................
000000d0: 7800 4000 0000 0000 0000 0000 0000 0000  x.@.............
000000e0: 0100 0000 0400 f1ff 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0300 0100  ................
00000100: a200 4000 0000 0000 0000 0000 0000 0000  ..@.............
```

….



1

Number Guessing Game

2

Note:

All the topics explained in the slideshow will be first taught, then shown in repl.it

3

Agenda

- Flow of game
- If statements
- Logic orders
- Random Numbers
- String operations
- Loops

4

1. Flow of game

How does the game work?

5

Flowchart—

A way to show the flow of an application.

6

Flowchart of Number Guessing Game

7

2. If statements

What if?

8

What are if statements?

- If statements are statements put into a code to check if a given condition is true as seen here:
  if(input()=="hello"):
      print("Hello! ")
- This will print hello if you say hello.

9

What are if statements?

- Using an else statement after the if statement will continue the code to that section, only if the if statement is not true.
- "elif", or else if, is used for multiple conditions.

10

Comparisons & booleans

- Booleans are basically conditions that you can check if something is true
- There are a number of comparison signs to compare different numbers and strings.
- The indent is REQUIRED, as Python is indent-sensitive.

11

Comparisons & booleans

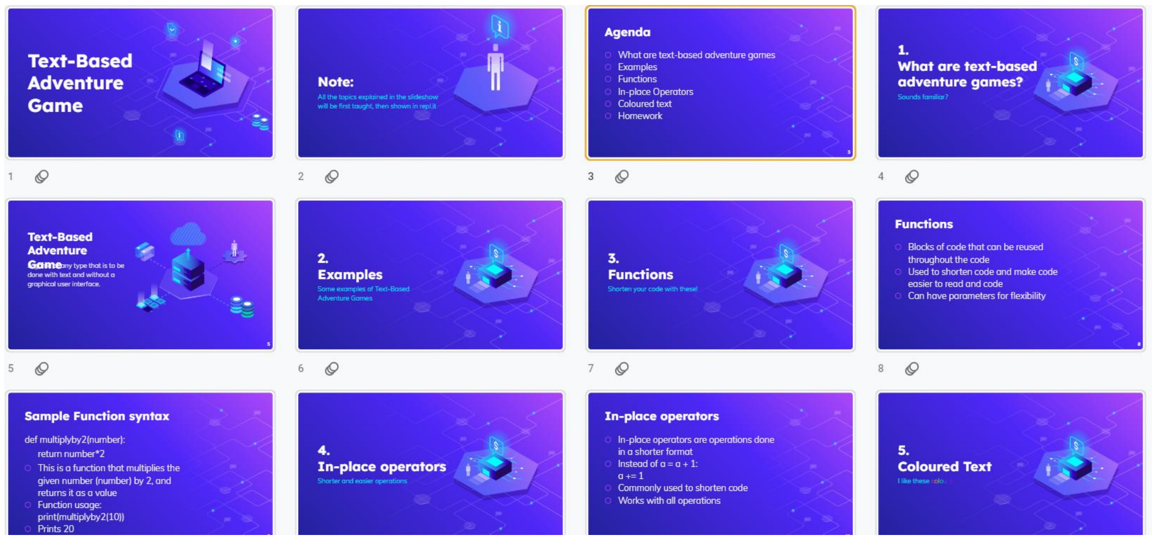
- To compare numbers, you can use the following signs—
  - > - Greater than
  - < - Less Than
  - == - Equal to
  - != - Not equal to
  - >= - Greater than or Equal to
  - <= - Less than or Equal to

12

Comparisons & booleans

- An example use of comparisons shows as follows:
  if(1 + 1 == 2):
      print("1 + 1 indeed equals to 2")
- Additionally, you can put a not or ! in front of a boolean to "invert" the outcome.
- You can also use == and != with strings to compare the strings.

…



1

Text-Based Adventure Game

2

Note:

All the topics explained in the slideshow will be first taught, then shown in repl.it

3

Agenda

- What are text-based adventure games
- Examples
- Functions
- In-place Operators
- Coloured text
- Homework

4

1. What are text-based adventure games?

Sounds familiar?

5

Text-Based Adventure Game

Any type that is to be done with text and without a graphical user interface.

6

2. Examples

Some examples of Text-Based Adventure Games

7

3. Functions

Shorten your code with these!

8

Functions

- Blocks of code that can be reused throughout the code
- Used to shorten code and make code easier to read and code
- Can have parameters for flexibility

9

Sample Function syntax

def multiplyby2(number):
    return number*2
- This is a function that multiplies the given number (number) by 2, and returns it as a value
- Function usage:
  print(multiplyby2(10))
- Prints 20

10

4. In-place operators

Shorter and easier operations

11

In-place operators

- In-place operators are operations done in a shorter format
- Instead of a = a + 1:
  a += 1
- Commonly used to shorten code
- Works with all operations

12

5. Coloured Text

I like these colo

…

Grade 4/5 Coding Lesson Plan (Based on Coding is Fun's beginner class lesson 1)

- What will we do?
  - Open a store
  - Build a billing program

- Objective
  - Understand the basics of the computer programming using python language

- Class schedule
  - Day 1: open a store
  - Day 2: take costumer's order
  - Day 3: calculate the bill
  - Day 4: turtle drawing or live help
  - Day 5: showcase
  - (if it's a four-day event then choose one from day 4 and day 5)

- Agenda (45 mins per class)
- Day 1: open a store
  - Introduce yourself: 10 mins
  - Design thinking: 5 mins
  - Brainstorming for a store idea: 10 mins
  - Brainstorming for product catalog: 5 mins
  - Translate from English to Python: print(): 5 mins
  - Exercise: 5 mins
  - Conclusion and homework: 5 mins

- Day 2: take customer's order
  - What we learned at last class: 5 mins
  - Present your homework: 2 students: 5 mins
  - Brainstorming: what will you do when your customer comes to your store? 5 mins
  - Concept of taking an order in English: 5 mins
  - Translate from English to Python: input(): 5 mins
  - Exercise: 5 mins
  - Check exercise: 10 mins
  - Conclusion and homework: 5 mins

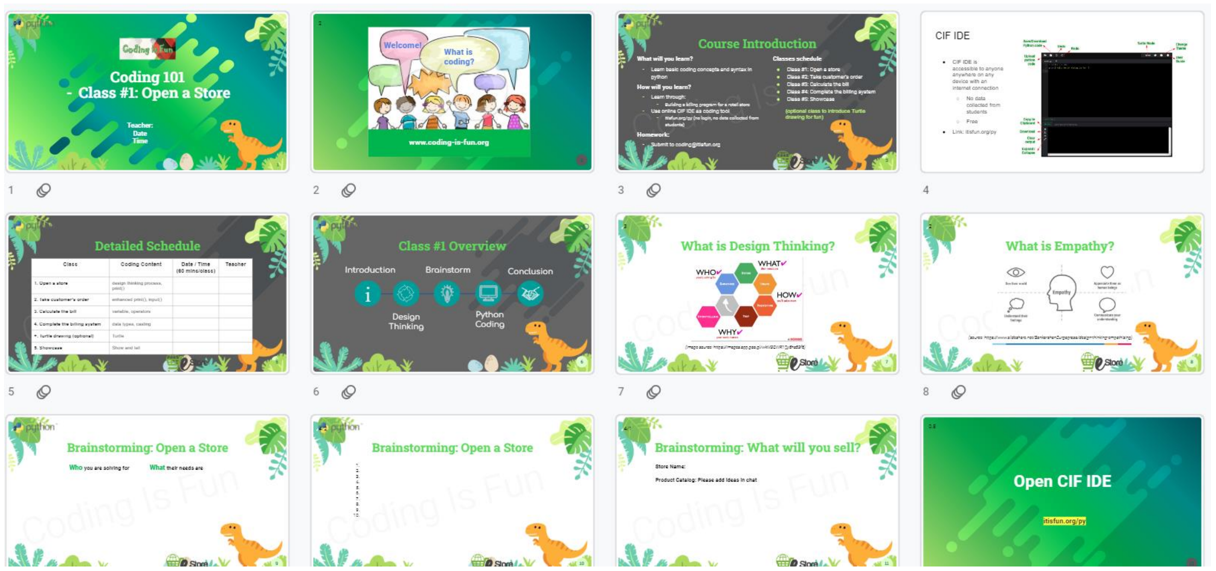
- Day 3: calculate the bill
  - What we learned at last class: 5 mins
  - Present your homework: 2 students: 5 mins
  - Brainstorming: how to calculate a bill? 5 mins
  - Concept of data type in English: 5 mins
  - Translate from English to Python: operators and casting: 5 mins
  - Exercise: 5 mins

  - Check exercise: 10 mins
  - Conclusion and homework: 5 mins

- Day 4:
  - What we learned at last class: 5 mins
  - Present your homework: 2 students: 5 mins
  - Depending on students' performance. If the homework goes well, then we teach turtle; otherwise do live help

- Day 5: showcase
  - Students' presentations
  - Teachers' presentations to inspire kids (show what python can do: drawing, gaming, maps, face recognition, etc.)

- Link to the Slides used



…