From Last Time…

- The Church-Turing Thesis tells us that any computer program you can write in your favorite language can also be implemented in a Turing Machine (and vice-versa)

Programs that Use Programs as Input

- Did you know that in Java, we can create programs that:
  - Take C code as input, and convert it into equivalent Python code?
  - Take C code as input, and actually check that it is syntactically correct and could run properly?
  - Take C code as input, and actually simulate the execution of the program?

Programs that Use Programs as Input

- Did you know that in Java, we can create programs that:
  - Take C code as input, and convert it into equivalent Python code? -- Translators
  - Take C code as input, and actually check that it is syntactically correct and could run properly? – Compilers/Typecheckers
  - Take C code as input, and actually simulate the execution of the program? -- Interpreters/VMs

… and furthermore, Church-Turing tells us that we can do all of these things, no matter what languages we choose!

Programs that Use Programs as Input

- Did you know that in ~~Java,~~ C we can create programs that:
  - Take C code as input, and convert it into equivalent Python code?
  - Take C code as input, and actually check that it is syntactically correct and could run properly?
  - Take C code as input, and actually simulate the execution of the program?

Programs that Use Programs as Input

- Did you know that ~~in Java,~~ we can create Turing Machines that:
  - Take C code as input, and convert it into equivalent Python code?
  - Take C code as input, and actually check that it is syntactically correct and could run properly?
  - Take C code as input, and actually simulate the execution of the program?

Not only that, we can also write Turing Machine descriptions in the same way we write programs in other languages

(The TM Simulator website does this)

Programs that Use Programs as Input

- Did you know that ~~in Java,~~ we can create Turing Machines that:
  - Take a TM description as input, and convert it into equivalent Python code?
  - Take a TM description as input, and actually check that it is syntactically correct and could run properly?
  - Take a TM description as input, and actually simulate the execution of the program?

Universal Turing Machines

- A Universal Turing Machine is a TM that:
  - Takes the pair <M, w> as input, where M is a description of a TM, and w is an input string for M
  - Simulates M on input w, where it…
    - Accepts, if M accepts w
    - Rejects, if M rejects w
    - Loops forever, if M loops forever on input w

Using UTMs

- Universal Turing Machines are essentially used to “run” other Turing Machines (using their descriptions)
- UTMs are incredibly useful when we want to do constructions of other TMs that do exotic things -- like modify the code of another TM and then run it!