APS106

UNIVERSITY OF TORONTO

Variables, Expressions and Operators.

Week 1 | Lecture 2 (1.2)

if nothing else, write #cleancode

This Week's Content

- **Lecture 1.1**
  - Introduction
- **Lecture 1.2**
  - **Variables, Expressions, and Operators**
  - **Chapters 1,2, and 3**

Lecture 1.3

- The Programming Process

Recap: What is Programming?

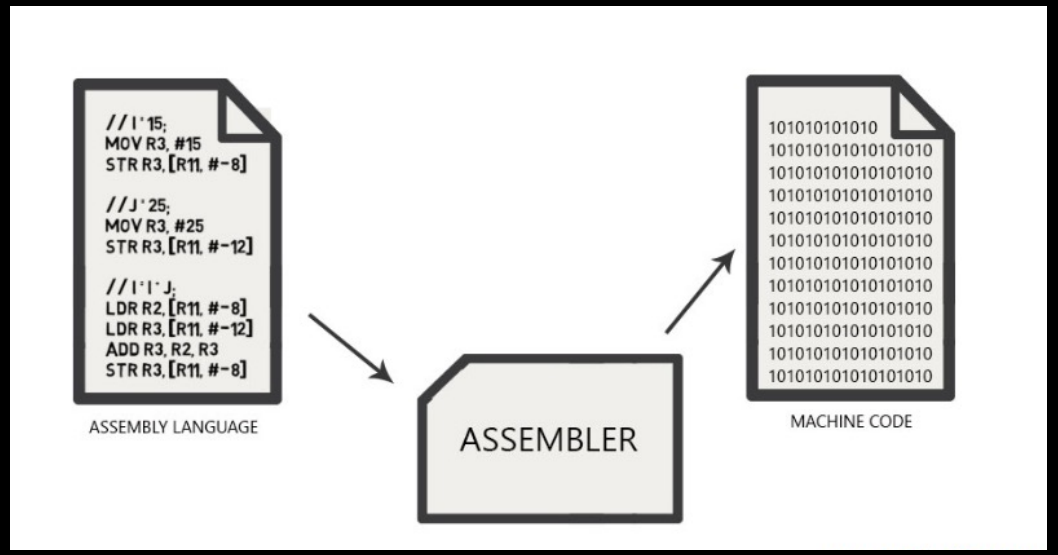
- A way of telling a computer what to do.
- A computer can't infer (…yet).
  - Need to tell a computer every single step it needs to do in a language it can understand.
  - How would you request an egg for breakfast to a chef and to a computer/robot?

- **To a Chef**
  1. Sunny-side up, please!

- **To a Computer**
  1. "Turn on stove"
  2. "Take out pan"
  3. "Take one egg out of fridge"
  4. "Crack egg"
  5. "Pour egg into pan"
  6. "Wait 5 minutes"

How to Program a Computer.

- `if x > 10:`
  `print("x is greater than 10")`



Programmer

Compiler

Assembler

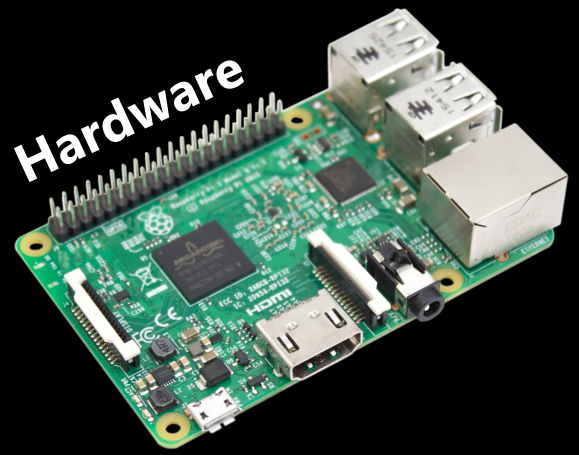
English

Pseudocode

Programing Language

Assembly Code

Machine Code



The power of programming languages

- `if x > 10:`

  `print("x is greater than 10")`

```
    pushl   %ebp                # \
    movl    %esp, %ebp          #  ) reserve space for local variables
    subl    $16, %esp           # /
    call    getint              # read
    movl    %eax, -8(%ebp)      # store i
    call    getint              # read
    movl    %eax, -12(%ebp)     # store j
A:  movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    cmpl    %ebx, %edi          # compare
    je      D                   # jump if i == j
    movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    cmpl    %ebx, %edi          # compare
    jle     B                   # jump if i < j
    movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    subl    %ebx, %edi          # i = i - j
    movl    %edi, -8(%ebp)      # store i
    jmp     C
B:  movl    -12(%ebp), %edi     # load j
    movl    -8(%ebp), %ebx      # load i
    subl    %ebx, %edi          # j = j - i
    movl    %edi, -12(%ebp)     # store j
C:  jmp     A
D:  movl    -8(%ebp), %ebx      # load i
    push    %ebx                # push i (pass to putint)
    call    putint              # write
    addl    $4, %esp            # pop i
    leave                       # deallocate space for local variables
    mov     $0, %eax            # exit status for program
    ret                         # return to operating system
```

Introducing Python

- No end-of-instruction separators, such as semicolons (like in C or Java)
- **Programs are stored in .py files**
- Comments start with a # character
- **Whitespace matters (exactly 4 spaces means indentation)**
- Python is an interpreted language (not a compiled one)
  - You can run code one statement at a time, just like a calculator or Matlab
  - This means variables can change type during runtime, and do not have to be declared before running

Arithmetic Operators

Operator	Operation	Expression	English description	Result
`+`	addition	`11 + 56`	11 plus 56	`67`
`-`	subtraction	`23 - 52`	23 minus 52	`-29`
`*`	multiplication	`4 * 5`	4 multiplied by 5	`20`
`**`	exponentiation	`2 ** 5`	2 to the power of 5	`32`
`/`	division	`9 / 2`	9 divided by 2	`4.5`
`//`	integer division	`9 // 2`	9 divided by 2	`4`
`%`	modulo (remainder)	`9 % 2`	9 mod 2	`1`

Arithmetic Operator Precedence

- When multiple operators are combined in a single expression, the operations are evaluated in order of precedence (from left to right)

Operator	Precedence
**	highest
- (negation)	
*, /, //, %	
+ (addition), - (subtraction)	lowest

Using Python as a Calculator

- Let's start with a simple use of Python to get the feel for our new environment
- Don't forget your BEDMAS (or PEMDAS)!

Open your notebook

Click Link:
1. Using Python as a Calculator

Variables and Memory

- The most basic thing you can do in a computer program is to assign a value to a variable.
- Assignment statements

variable = expression

  - x = 20

    (or)
  - y = 20+5*2

- Rules for assignment
  - 1. Evaluate the expression to the right of = sign (produces memory address of the value)
  - 2. Store the memory address in the variable on the left of the = sign

Variable Names and Conventions

- The rules for legal Python names:
  - Names must start with a letter or _ (underscore)
  - Names must contain only letters, digits, and _
- In most situations, the convention is to use pothole_case
  - Lowercase letters with words separated by _ to improve readability
- Try to add meaning where possible!
  - Ex: gas_mileage and cost_per_litre instead of nomnom and nomnomnom
  - Save yourself when debugging & put your TAs in a good mood when marking

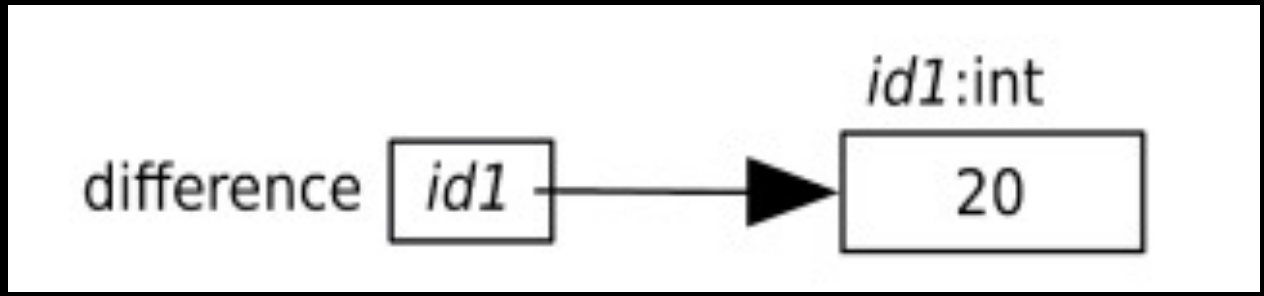
Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

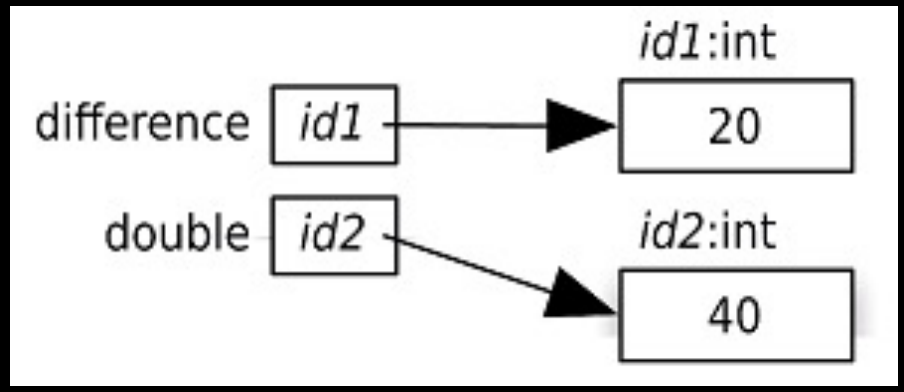
1. Evaluate the expression on the right of the = sign -> 20. This produces the value 20, which we'll put at memory address id1.
2. Make the variable on the left of the = sign, difference, refer to 20 by storing id1 in difference.



Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

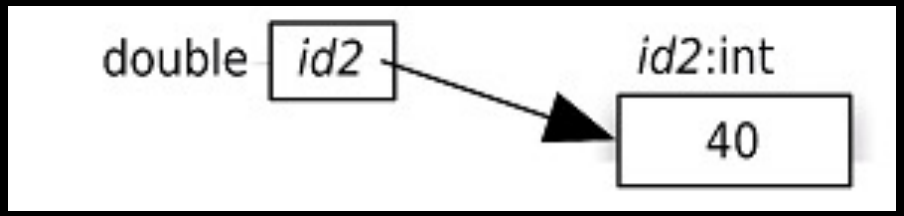
1. Evaluate the expression on the right of the = sign: 2 * difference. As we see in the memory model, difference refers to the value 20, so this expression is equivalent to 2 *20, which produces 40. We'll pick the memory address id2 for the value 40.
2. Make the variable on the left of the = sign, double, refer to 40 by storing id2 in double.



Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

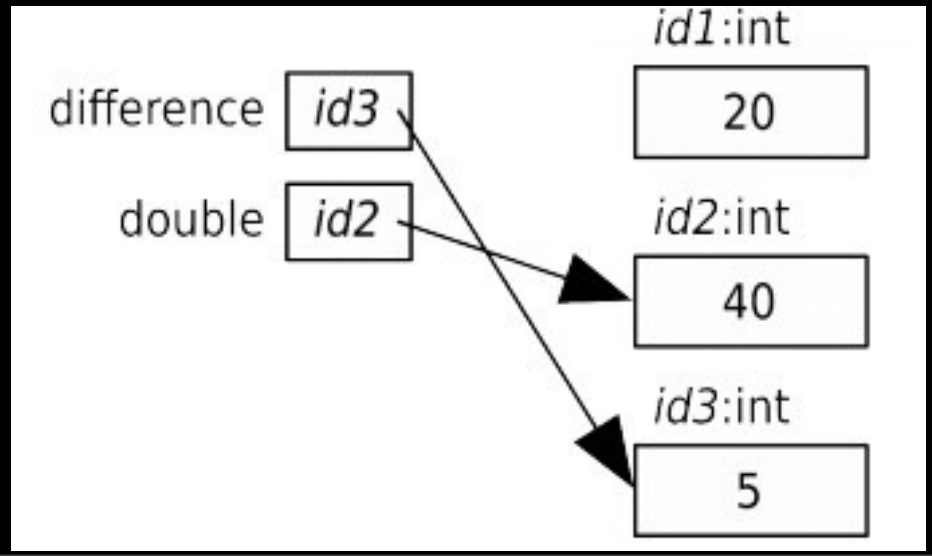
When Python executes the third statement, double, it merely looks up the value that doublerefers to (40) and displays it.



Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

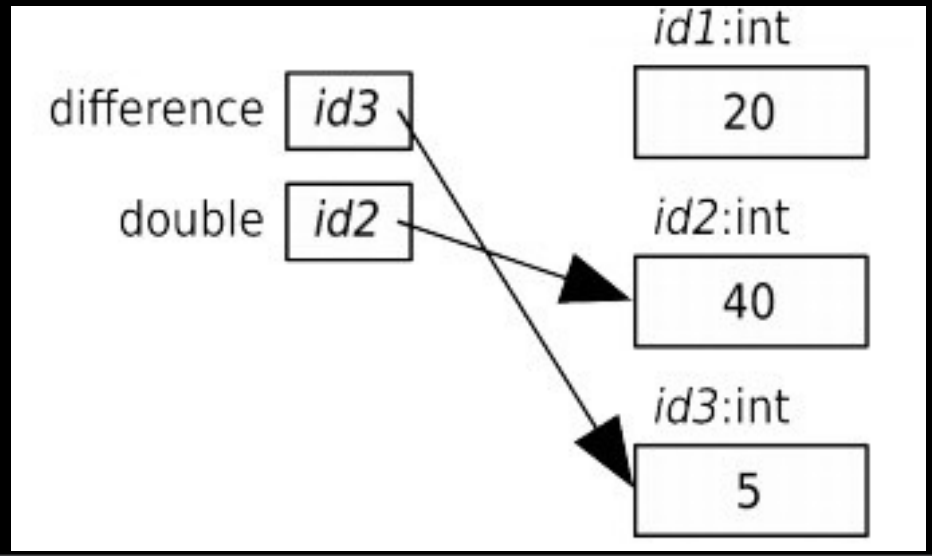
1. Evaluate the expression on the right of the = sign: 5. This produces the value 5, which we'll put at the memory address id3.
2. Make the variable on the left of the = sign, difference, refer to 5 by storing id3 in difference.



Memory Visualization Example

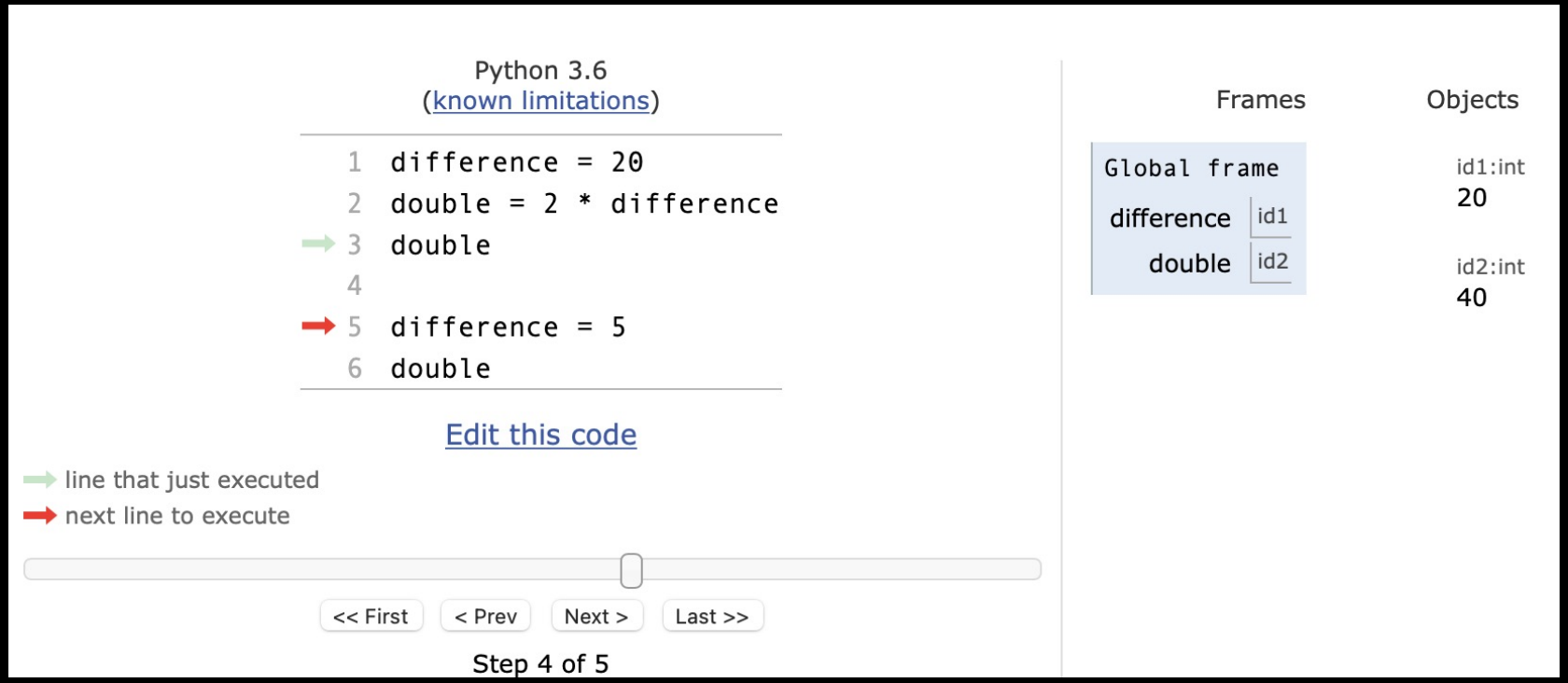
```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```

Variable double still contains id2, so it still refers to 40. Neither variable refers to 20 anymore which is OK as Python will take care of recycling of memory when it is no longer used.



Python Tutor Visualization

- http://pythontutor.com/visualize.html
- Code might seem simple now…



Variables and Memory

- Let's go experiment with some of what we just saw
  - Assignment statements
  - Declaring variables
  - Different variable names
  - Memory locations
  - (and the famous print statement!)

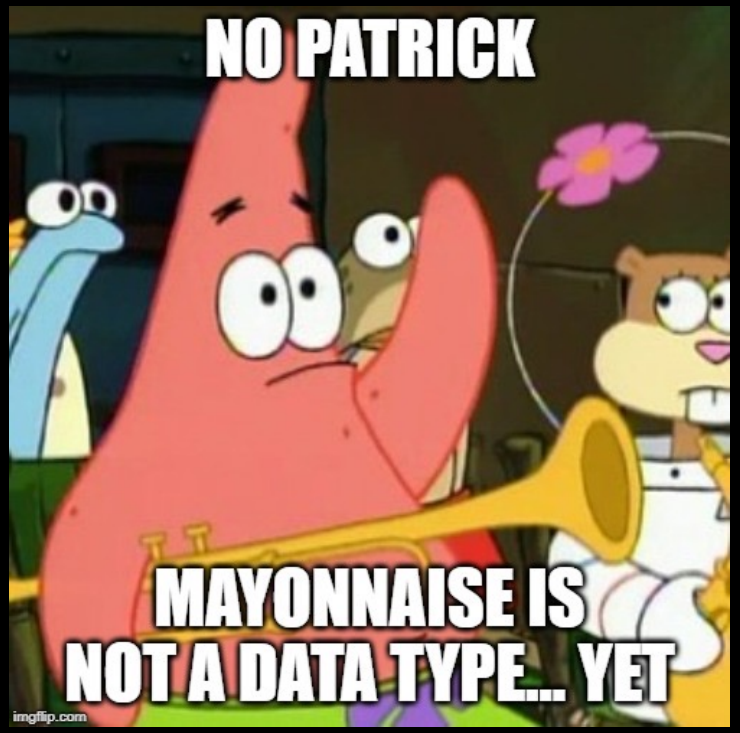
Open your notebook

Click Link:
2. Variables and Memory

Variable Types

- A **type** is a set of *values* and the *operations* that can be performed on those values.

- *int*: integer
  - ex. 3, 4, 894, 0, -3, -18
- *float*: floating point number
  - ex. -5.6, 7.342, 53452.0, -89.34



Numerical Type Examples

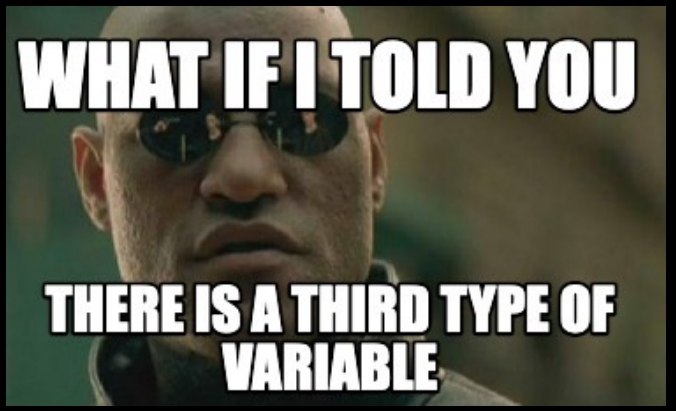
```
>>> 5 + 2 * 4

>>> 5.0 + 2 * 4

>>> 30/6

>>> 30//6
```

Type: str (pronounced string)



- *str:* string literal is a sequence of characters
- Start and end with single quotes (') or double quotes (")
  - ex. 'hello', "What is 10 * (2 + 9)?"
  - Just like writing in English, the quote type must match (i.e. 2 singles or 2 doubles, not 1 of each)

String Type Examples

```
>>> 'how are you?'

>>> "short- and long-term"

>>> '"APS106" is already my favourite course'

>>> "APS106 stinks'
SyntaxError
```

More data types?!

- You can get the data type of any object by using the type() function

```
>>> x = 5
>>> print(type(x))
<class 'int'>
```

- Other Python data types:
  - List
  - Tuple
  - Dictionary
  - Set
  - Boolean

BUT we will worry about these later…

Variables Types

- Let's go experiment with some of what we just saw
  - "int"s vs "float"s
  - The "str" type
  - Using types and variables in expressions
  - Combining our type knowledge with some arithmetic operations

Open your notebook

Click Link:
3. Different Types of Variables

Augmented Assignment Operations

Operator	Expression	Identical Expression	English description
+=	x = 7 x += 2	x = 7 x = x + 2	x refers to 9
-=	x = 7 x -= 2	x = 7 x = x - 2	x refers to 5
*=	x = 7 x *= 2	x = 7 x = x * 2	x refers to 14
/=	x = 7 x /= 2	x = 7 x = x / 2	x refers to 3.5
//=	x = 7 x //= 2	x = 7 x = x // 2	x refers to 3
%=	x = 7 x %= 2	x = 7 x = x % 2	x refers to 1
**=	x = 7 x **= 2	x = 7 x = x ** 2	x refers to 49

Open your notebook

Click Link:
4. Augmented Assignment Operators

Let's Code!

Convert gas mileage from American to Canadian

- In the old days (and still in the United States), the mileage of a gas-powered car was measured in miles per gallon.
- Now for places that use the metric system, we prefer to measure "mileage" as "fuel consumption" in litres per hundred kilometres.
- Write code to do the conversion to metric given a value in miles per gallon.

Open your notebook

Click Link:
5. Let's Code!

APS106

UNIVERSITY OF TORONTO

Variables, Expressions and Operators.

Week 1 | Lecture 2 (1.2)

if nothing else, write #cleancode