Module 2: Variables

- Now we learned what programming languages are, and why we use Python.
- We also learned how to use print() and input() to build interactive programs for simple input/output.
- It is not hard, but,
- It is not smart, right?
- We need variables!

Variables

- What are variables?
  - You can imagine a variable is a reference of a data.
  - When you want to compute the data, you can use the variable instead.
    - When you use "Capitol Hill" in a conversation, people will understand you are referring to the specific building in Washington, D.C.
    - When you use x in a computation, Python will use the data that x is referring.

Identifiers of Variables

- Variables have identifiers - their names.
  - Each identifier may consist of letters (A-Z, a-z), digits (0-9), and underscores (_).
  - Identifiers must begin with a letter or the underscore.
  - Identifiers may not begin with a digit.
  - Identifiers cannot have special characters ($, %, #, &, *, ^, etc)
  - Identifiers are case sensitive.
  - Identifiers may not be reserved keywords
    - Such as import, if, else, and, or, return

Naming Conventions

- Naming variables is not just for correct.
- We follow naming conventions for readability.
  - For our course, most of the case, we name a variable with lower case letter.
    - name, age, x, location, income, etc
  - The name should be meaningful so readers can understand.
  - You can connect multiple words use underscore, or capitalize the 1st letter from second words.
    - average_income, start_location, first_name
    - totalSale, profitByQuarter, lastName

Assignment Operation

- We use = to assign a value to a variable.
  - Variable name on the left of =
  - Value on the right of =
  - This single equation is called Assignment Operation.
- message = "Hello, world!"
  - A value "Hello, world!" is assigned to variable message.
  - print(message) will be the same as print("Hello, world!")

Type of Variables

- The type of a variable is determined by the type of current data the variable is referring.
  - type(x) returns the type of current data variable x refers.
  - Let's see what the data type of "Hello, world!" is.
  - It is a 'str': String.
  - What is the type string?

String

- String is a predefined data type in Python.
- It is a sequence of characters.
- Single line string is wrapped with ' ' or " ".
  - 'Hello, world!'
  - "Hello, world!"
- Multiline string is wrapped with ''' ''' or """ """.
  - '''Hello
    world'''

Numeric Types

- To represent numeric type, Python has:
  - int: a numeric value without decimal points:
    - 1, -1, 20, 9999999999999999, etc.
  - float: a numeric value has decimal points:
    - 1.0, -1.1, 20.1, 9.9999999999, etc.

Dynamic Typed Variables

- Python is a dynamic-type programming language.
  - Variable that has been assigned to one type can be reassigned to another type.
    - x = "some text", now x has type as string.
    - We can assign x = 6, now x has type as int.
    - We can assign x again x = 5.9, now x has type as float.

Value Can Be Transferred

- A variable that refers to a value can be on the right side of assignment too:
  - x = 5, x is now referring to int 5
  - y = x, y is now referring to int 5 too.
- Only value will be transferred:
  - If we assign x with a different value, say x = 6
  - y is still referring to int 5.

Value Can Be Converted

- A valid value may be converted to other data type.
  - From str/float to int:
    - x = int('5'), x will refer to int 5.
    - X = int('A') will raise an error.
    - x = int(5.0), x will refer to int 5.
    - X = int(5.5), x will refer to int 5. (Value after decimal point will be discarded)
  - From int/float to str:
    - x = str(5), x will refer to str '5'.
  - From int/str to float:
    - x = float(5), x will refer to float 5.0.

Let’s Play in Colab

- Download the M2Lab1.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Arithmetic Operations

- Numeric value can be computed with arithmetic operations:
  - + Addition: Adds two numbers
  - – Subtraction: Subtracts one number from another
  - * Multiplication: Multiplies one number by another
  - / Division: Divides one number by another and gives the result as float

Arithmetic Operations

- Numeric value can be computed with arithmetic operations:
  - // Integer division: Divides one number by another and gives the result as int
  - % Remainder: Divides one number by another and gives the remainder
  - ** Exponent: Raises a number to a power

Precedence

- The arithmetic operations has following precedence:
  - Highest: ()
  - Second: **
  - Third: *, /, //, %
  - Lowest: +, -
- What is the answer of the operation below:
  - ((312-213)*(29.2-2)+33%3-12/3)*(10//5-2)

input() Revisit

- We learned input() to get input from users.
- What is the data type of the result of input()?
  - type(input('Enter some text:'))
  - type(input('Enter an integer:'))
  - type(Input('Enter a float:'))
  - input() will always give you a str.

input() Revisit

- You can converter the value from input():
  - int(input('Enter your age:'))
  - float(input('Enter your GPA':))
  - Errors will be raised if the user entered something not valid for converting.
- You can get multiple inputs at a time:
  - first_name, last_name = input('What is your first name and last name? Separate by a comma:').split(',')

Let’s Play in Colab

- Download the M2Lab2.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Relational Operations

- Very often we may need to compare two values.
  - We use Relational Operations, so called Boolean expressions:

Expression	Meaning
x > y	Is x greater than y?
x < y	Is x less than y?
x >= y	Is x greater than or equal to y?
x <= y	Is x less than or equal to y?
x == y	Is x equal to y?
x != y	Is x not equal to y?

Boolean Type

- The result of a relational operation will be a boolean type: bool.
  - bool is a predefined data type in Python.
  - Only two values: True, False

Logical Operations

- Boolean type can be computed by Logical Operations.
  - and, or, not.
  - and operator and or operator are binary operators, they need to connect two boolean expressions.
  - not operator is a unary operator, reverses the value.
  - Truth table shows the result of logical operations.

The `and` Operator

- Takes two Boolean expressions as operands
  - Creates compound Boolean expression that is true only when both sub expressions are true
  - Can be used to simplify nested decision structures
- Truth table for the and operator

Expression	Value of and operation
False and False	False
False and True	False
True and False	False
True and True	True

The `or` Operator

- Takes two Boolean expressions as operands
  - Creates compound Boolean expression that is true when either of the sub expressions is true
  - Can be used to simplify nested decision structures
- Truth table for the or operator

Expression	Value of or operation
False and False	False
False and True	True
True and False	True
True and True	True

Short-Circuit Evaluation

- Short circuit evaluation: deciding the value of a compound Boolean expression after evaluating only one sub expression
  - Performed by the or and and operators
    - For or operator: If left operand is True, compound expression is True. Otherwise, evaluate right operand
    - For and operator: If left operand is False, compound expression is False. Otherwise, evaluate right operand

The `not` Operator

- Takes one Boolean expressions as operand and reverses its logical value
  - Sometimes it may be necessary to place parentheses around an expression to clarify to what you are applying the not operator
- Truth table for the not operator

Expression	Value of not operation
not True	False
not False	True

Let’s Play in Colab

- Download the M2Lab3.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Use the discussion board to ask for help.

Let’s Play in Colab

- Download the M2Assignment.ipynb file.
- Upload it to your Colab.
- Finish the tasks.
- Submit your assignment.
- Use the discussion board to ask for help.

Congratulations!

- You finished Module 2.
- Programming in Python is still not that hard, right?
- We are going to learn more in next module and get even more power unlocked!
- See you soon!