Topic 2: Introduction to Programming

1

Recommended Readings

STARTING OUT WITH PYTHON®

TONY GADDIS

- Chapter 2

2

Computer Programming

- Gain necessary knowledge of the problem domain
- Analyze the problem, breaking it into pieces
  - Repeat as necessary
- Synthesize a solution
- Run the program
- Validate program results
  - Correct problems that are identified

3

Programming Languages

- Many programming languages available
  - Offer different features
  - Each has its own strengths and weaknesses

- Common features
  - Allow us to <u>control the behavior</u> of a computer
  - Defined syntactic and semantic rules

4

Levels of Abstraction

- Human Languages

- High Level Programming Languages

- Low Level Programming Languages

- Machine Language

5

Python

- A high-level general purpose programming language
  - Reasonably simple, easy to learn
  - Reasonably easy to find and fix program errors
  - Available for many platforms
  - Powerful enough to solve interesting problems
  - Used in industry (eg. Civ4, YouTube)

6

Programming

- Computer programs are stored in source files
  - typically have the extension .py

- Once the file is created, it is run using the python interpreter
  - `python myfile.py`

7

A First Python Program

- Write a Python program that converts a pressure from kilopascals into
  - atmospheres
  - pounds per square inch
  - millimetres of mercury

8

A First Python Program

- What steps can we follow to reach this goal?

9

Variables

- Variable
  - A <u>named</u> location in memory
  - Holds a value
  - The programmer can
    - read the value of a variable without changing / destroying the value
    - change the <u>value</u> of the variable
    - change the <u>type</u> of information stored in the variable

10

Variable Names

- Variable names
  - should be meaningful
  - must begin with a letter or an underscore
  - may contain a mixture of letters, numbers and underscores
  - must not be a reserved word
  - shouldn't be a name already commonly used for another purpose
  - shouldn't be in all caps

11

Assignment

- A variable is created and given a value using an <u>assignment statement</u>
  - The variable that gets a value appears to the left of the assignment operator
  - An arbitrarily complex expression appears to the right of the assignment operator
    - Expression may include other variables

12

Getting Input

- Python includes a library of functions that perform useful tasks
  - Our program can use these functions
  - A function is "called" by using its name
  - The function name is always followed by round brackets
    - May include values inside the brackets that are used by the function
  - Function result can be stored in a variable
  - Use `input()` to read a value

13

Generating Output

- Use a print statement
  - Behaves much like a function
    - Exception: No brackets
  - Can print numbers, strings, contents of variables, …
  - Multiple items can be printed
    - Separate each item with a comma

14

The Code

- In a file named Pressure.py:

15

Running the Program

- CPU can <u>only</u> execute machine language instructions
  - Can't execute programming language statements directly
  - Options:
    - Compile the program into machine language instructions
    - Use a Virtual Machine that reads your program and performs the tasks required to run it

16

Compilation

17

Virtual Machine

18

Comments

- Provide information to someone reading your code
  - Completely ignored by the computer
  - Should explain <u>how</u> or <u>why</u>
  - Should <u>add value</u>
    - A comment that says something that is immediately obvious from reading the code is not particularly useful

19

Magic Numbers

- Magic Number: An <u>unnamed</u> and/or <u>poorly documented</u> numeric constant without obvious meaning
  - Should be avoided
    - Program is difficult to understand
    - Errors are difficult to detect
    - If the value changes, it may need to be changed in many places

20

What Does this Program Do?

```
x = input()
y = 32 + x * 9/5
print y
```

- What's wrong with this program?

21

Expressions

- Python supports arbitrarily complex mathematical expressions
  - Integers / Floating Point Numbers / Parentheses
  - Operators
    - +: addition
    - -: subtraction
    - *: multiplication
    - /: division
    - %: remainder
    - **: exponentiation

22

Precedence

- The order evaluation is determined by operator precedence
  - ()
  - -x, x**y
  - x*y, x/y, x%y
  - x+y, x-y
  - =
  - Evaluation is left to right at each level

23

Example

24

Math Functions

- Many additional math functions are available
  - Located in the math library
    - Import the math library
    - Precede the name of the function with math.
  - Examples:
    - math.sqrt(x)
    - math.floor(x)
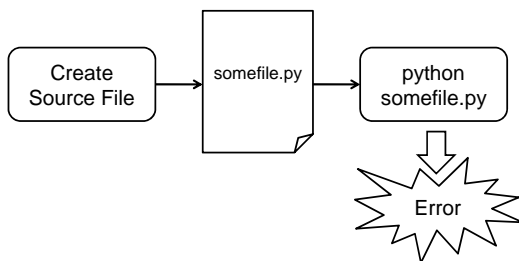   - math.ceil(x)
    - math.cos(x)

25

Types of Errors

- Three categories of errors:
  - Syntax Errors
  - Runtime Errors
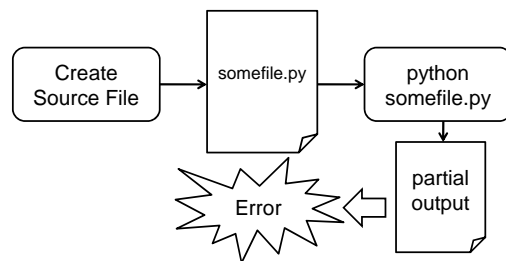 - Logic Errors

26

Syntax Errors

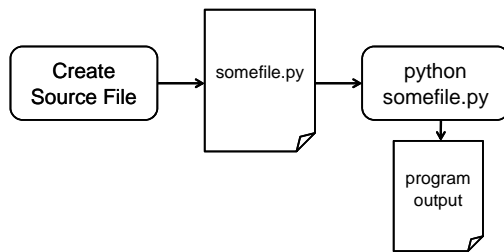- Identified as code is loaded
- No statements are executed

| Create Source File | → | somefile.py | → | python somefile.py |

Error

27

Runtime Errors

- Identified as the program runs
- Program does not complete successfully

| Create Source File | → | somefile.py | → | python somefile.py |

partial output

Error

28

Logic Errors

- Program runs to completion, but generates incorrect results

```
Create
Source File  →  somefile.py  →  python
                                somefile.py
                                     ↓
                               program
                               output
```

29

Types

- Variables hold values
  - Each value has a type
    - Integer
    - Float
    - Boolean
    - String
    - ...

30

Types

- Some operations are only well defined for certain types
  - 1 + 2 =
  - "Hello" + " World" =
  - 1 + "Hello" =
  - 2 + "4" =
  - 1 / 3 =
  - 2.0 / 4 =

31

Type Conversions

- Python permits you to "cast" from one type to another
  - "1.0" / "3.0" =
  - float("1.0") / float("3.0") =
  - float("asdf") =

- Other type casts: int, bool, str

32

Strings

- Reading Strings
  - input() function reads integers, floats and Booleans
    - Generates an error if used to read a string
  - raw_input() function reads strings
    - Value is stored as a string even if the characters form a number

33

String Methods

- Methods also available for
  - Formatting
  - Searching
  - Counting
  - Separating
  - Testing Properties
- Will discuss strings in detail later…

34

Example

- Consider getting a loan for a sports car
  - Want to compare payments for different
    - Amount borrowed
    - Interest rate (percentage per year)
    - Amortization period
  - Write a program that
    - reads the amount borrowed, interest rate and amortization period
    - Displays monthly payment & total borrowing cost

35

Example

- Useful Equation: $P = \dfrac{iA}{1-(1+i)^{-N}}$

  - P: Payment amount
  - i: Interest rate per payment period as a decimal value
    - 5% should be 0.05
  - A: Amount borrowed
  - N: Total number of payments

36

Example

37

Formatting Output

- Sometimes print doesn't display things the way we would like
  - print 1 / 3.0 gives 0.333333333333
    - What if we want 0.33?
    - What if we want to center the result on the line?
    - What if we want to right-justify the result?

38

Formatting Numbers

- The % operator can be used to format numbers
  - Format specifier to its left
    - A string that controls how the value will be formatted
  - Expression that evaluates to a number on its right

  - Example: "%.2f" % 3.14159265

39

Format Specifiers

- A string
  - Format starts with a %
  - Number(s) and optional decimal point control formatting
  - Letter indicates type
    - f to format floating point numbers
    - d to format an integer in decimal format
    - x to format an integer in hexadecimal format

40

Wrapping Up

- Programming
  - Process of converting an algorithm to a form that can be executed by a computer
- A program
  - Uses variables to hold values
  - Evaluates expressions
  - Calls functions to get input, perform mathematical operations
  - Uses print statements to generate output

41

Where Are We Going

- Programs we can write are limited
  - What if we want different behaviour depending on a value entered by the user?
  - What if we want to perform a task several times?
- What kinds of data can a computer manipulate?
  - How does the computer represent data?

42