Topic 2: Introduction to Programming

Textbook

- Strongly Recommended Exercises
  - The Python Workbook, 2nd Edition: 12, 13, 23, and 29
- Recommended Exercises
  - The Python Workbook, 2nd Edition: 5, 7, 15, 21, 22 and 32
- Recommended Reading
  - The Python Workbook, 2nd Edition: Chapter 1

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

Programming Languages

- Many programming languages available
  - Offer different features
  - Each has its own strengths and weaknesses
- Common features
  - Allow us to control the behaviour of a computer
  - Defined syntactic and semantic rules

Levels of Abstraction

- Human Languages
- High Level Programming Languages
- Low Level Programming Languages
- Machine Language

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

Programming

- Computer programs are stored in source files
  - Human readable / editable
  - Can also be understood by a computer
  - Typically have the extension .py
- Once the file is created, it is run using the python interpreter

```
python myfile.py
```


A First Python Program

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

A First Python Program

- What steps can we follow to reach this goal?

Variables

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

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

Assignment

- A variable is created and given a value using an assignment statement
  - 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

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

Getting Input

- The input() function reads characters typed by the user as text
  - It normally appears on the right side of an assignment statement

```
name = input()
```

  - If we want to treat the characters read by the user as a number, we must perform a conversion

```
num = float(input())
```


Generating Output

- Use a print statement
  - It’s another function
  - Can print numbers, text, variables, …
  - Multiple items can be printed
    - Separate each item with a comma

The Code

- In a file named pressure.py:

Running the Program

- CPU can only 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

Comments

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

Magic Numbers

- Magic Number: An unnamed and/or poorly documented 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

What Does this Program Do?

```
x = float(input())
y = 32.0 + x * 9.0/5.0
print(y)
```

- What's wrong with this program?

Expressions

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

Precedence

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

Example

Math Functions

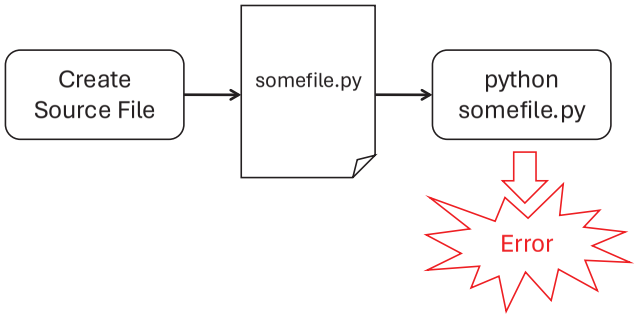
- Many additional math functions are available
  - Located in the math module
    - Import the math module
    - Precede the name of the function with math.
  - Examples:
    - math.sqrt(x)
    - math.floor(x)
    - math.ceil(x)
    - math.cos(x)

Types of Errors

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

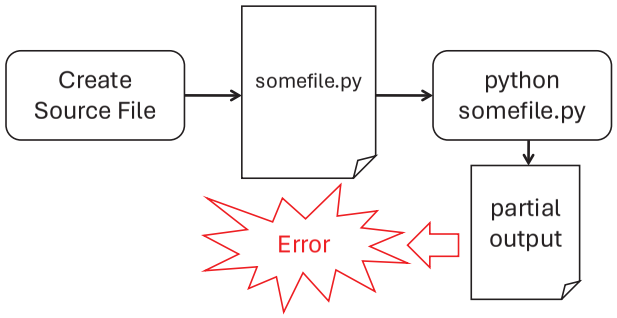
Syntax Errors

- Identified as code is loaded
- No statements are executed



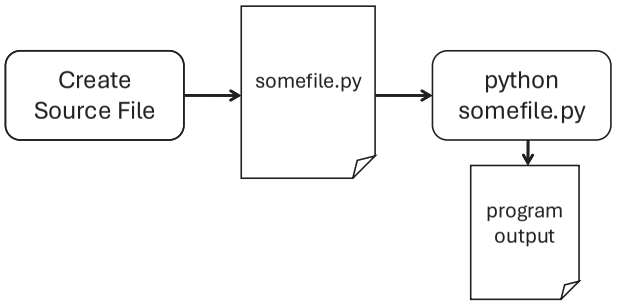
Runtime Errors

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



Logic Errors

- Program runs to completion, but generates incorrect results



Data Types

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

Data Types

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

Type Conversions

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

- Other type conversions: int, bool, str

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 and total borrowing cost

Example

- Useful Equation: $P = i A / (1 - (1 + i)^{-N})$
  - P: Payment amount
  - i: Interest rate per payment period as a decimal value
    - 5% should be 0.05, but the user will enter 5
  - A: Amount borrowed
  - N: Total number of payments

Example

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?

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

Format Specifiers

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

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
  - Calls the print function to generate output

Where Are We Going

- What kinds of data can a computer manipulate?
  - How does the computer represent data?
- 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?