A Brief Introduction to Programming in Python

M. Kumaresan & G. Leong

Mathematics Department, SHS

February 10, 2019

What is Programming?

- A program is a sequence of instructions that specifies how to perform a computation (mathematical or symbolic).
- Programming is the process of breaking a large, complex task into smaller and smaller subtasks until the subtasks are simple enough to be performed with basic instructions.

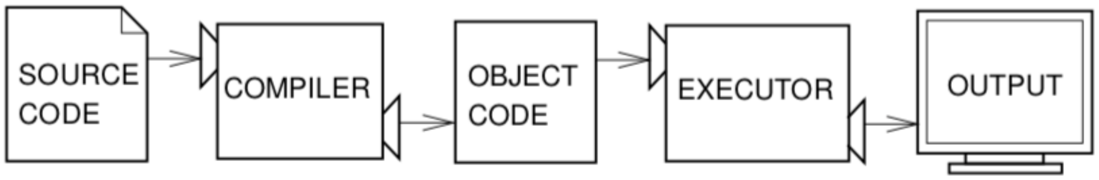
High Level vs. Low Level

Python is an example of a high-level language (other examples include C, C++, Java). There are low-level languages (“machine/assembly languages”) which are programs written for a specific piece of hardware. Computers can only execute programs written in low-level languages, thus programs written in high-level languages have to be processed before they can run. Here are two advantages to high-level languages:

1. Take less time to write, are shorter and easier to read, and are more likely to be correct.
2. Portable - can run on different kinds of computers with few/no modifications.

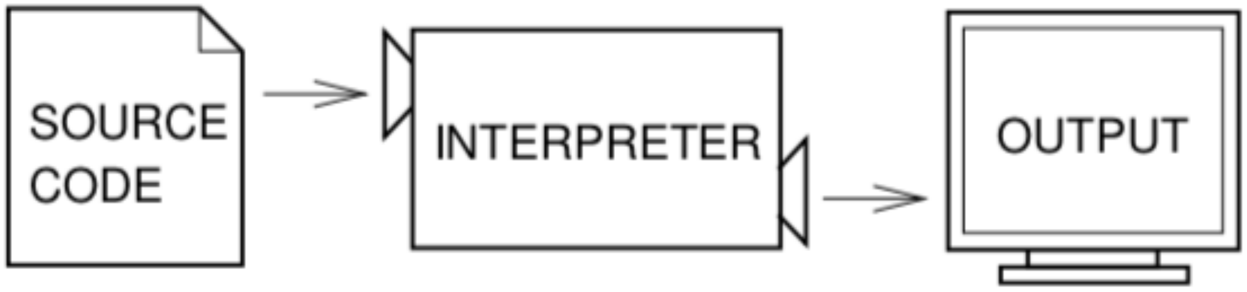
Compilers

A compiler reads the program and translates it completely before the program starts running. The high-level program is called the source code, and the translated program is called the object code/executable. Once a program is compiled, you can execute it repeatedly without further translation.



Interpreters

An interpreter reads a high-level program and executes it, meaning that it does what the program says. It processes the program a little at a time, alternately reading lines and performing computations.



Why Python?

- Python can be executed either as a compiled program, or as an interpreted language in interactive mode. We will go with the latter for simplicity and ease of learning. Interpreted languages give additional flexibility and features over compiled languages.
- Other benefits of Python:
  - Accessible (syntax, interpreted)
  - Balance between practical and conceptual
  - Powerful (large libraries of modules exist)
  - Industry-wide standard
  - Fun and more productive!

Numbers and Arithmetic

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Mod: %
- Exponents: **

Variables

Rules for Variable Assignment:

- Names cannot start with a number.
- There can be no spaces in the name (use underscore instead).
- Can't use any of the following symbols:
  :",<>/?|()!@#$%^&*~-+
- Best practice: use lowercase letters for variable names
- Avoid using words that have special meaning in Python like "list" and "str"
- When you run code, Python will tell you if you've made an error (SyntaxError)

Printing

- print(“string”)
- Float formatting follows: “{value:width.precision f}”

Exercises

Exercise

Write an equation that uses multiplication, division, an exponent, addition, the mod operator and subtraction that is equal to 100.25

Exercise

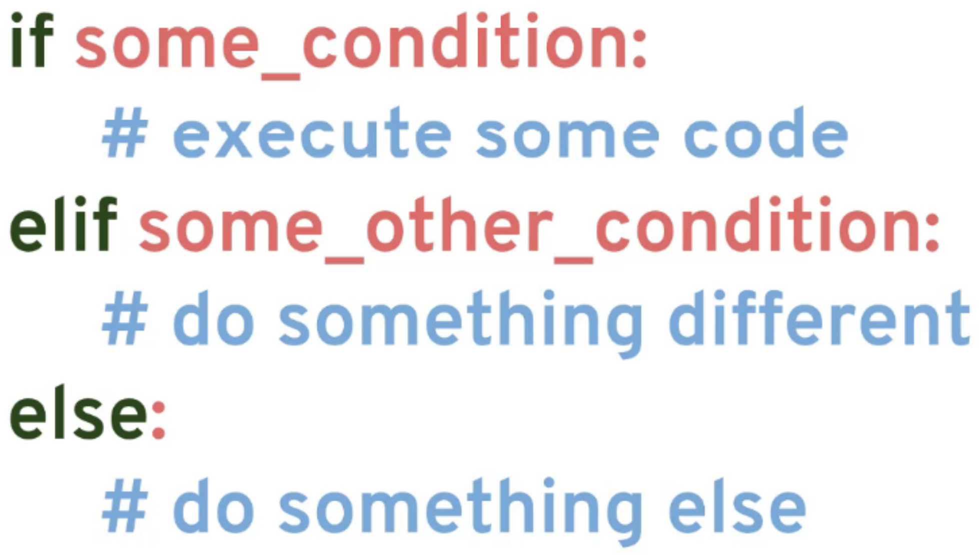
What is the type of the result $3 + 1.5 + 4$?

Exercise

Print the square root of two to 5 decimal places.

Branching

Control Flow syntax makes use of colons and indentations (whitespace). This indentation system is crucial to Python.



```
if some_condition:
    # execute some code
elif some_other_condition:
    # do something different
else:
    # do something else
```

Exercises

Exercise

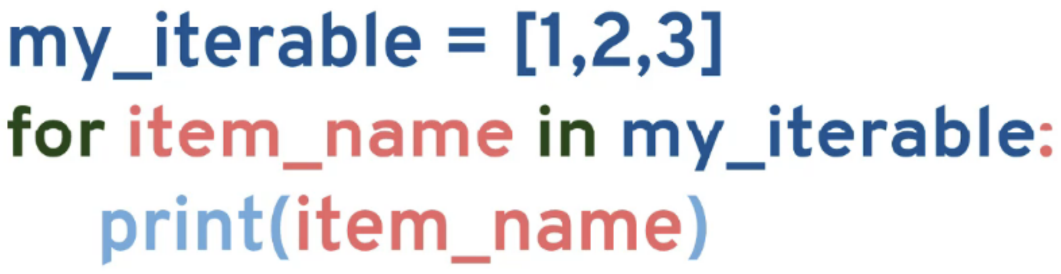
Given a user-generated integer, write code to determine whether the integer is even or odd. Print the result.

Exercise

Suppose Greg buys an item at his favorite store (Woodcraft). Write code to take the item amount from the user (Greg), and the amount tendered from the user. Determine the exact change owed to Greg in the largest denominations possible.

Looping

- We can use loops to iterate through certain "iterable" objects in Python (such as strings and lists). "For" loops are commonly used for iteration.



```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

- "break" allows you to exit out of a loop early if you are branching in a loop. it breaks out of the current closest enclosing loop.

Exercises

Exercise
Use **range**() to print all the even/odd numbers from 0 to 100.

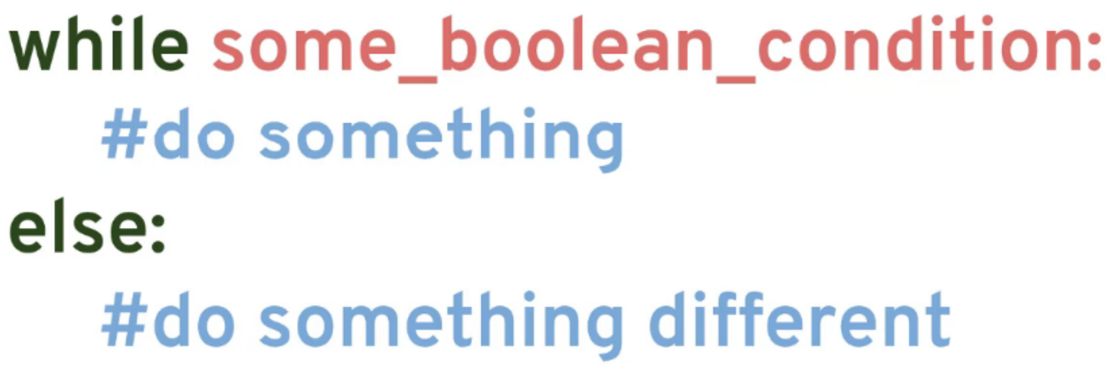
Exercise
Print the sum of the first 100 integers.

Exercise
Write a program that prints the integers from 1 to 100. But for multiples of three, print “Fizz” instead of the number, and for the multiples of five print “Buzz.” For numbers which are both multiples of three and five, print “FizzBuzz.”

Exercise
Print out all the prime numbers from 0 to 100.

While Loops

- Syntax:



```
while some_boolean_condition:
    #do something
else:
    #do something different
```

Exercises

Exercise

Given two user-generated integers, a and b, determine $\mathbf{gcd}(a, b)$.

Exercise

Given two fractions, add them and express the result in simplest form.