Introducción a las ciencias de la computación
y programación en Python

Introducción

Agenda del curso, evaluación, conceptos básicos, introducción a Python

Rodrigo Chang

Banco de Guatemala

Rodrigo Chang <rrcp@banguat.gob.gt>

Abstract

“Whether you want to uncover the secrets of the universe, or you just want to pursue a career in the 21st century, basic computer programming is an essential skill to learn.”
— Stephen Hawking

Discutiremos lo que aprenderemos en este curso y cómo lo vamos a evaluar.
Discutiremos algunos conceptos importantes en ciencias de la computación.
Hablaremos del lenguaje de programación Python y por qué lo vamos a utilizar.

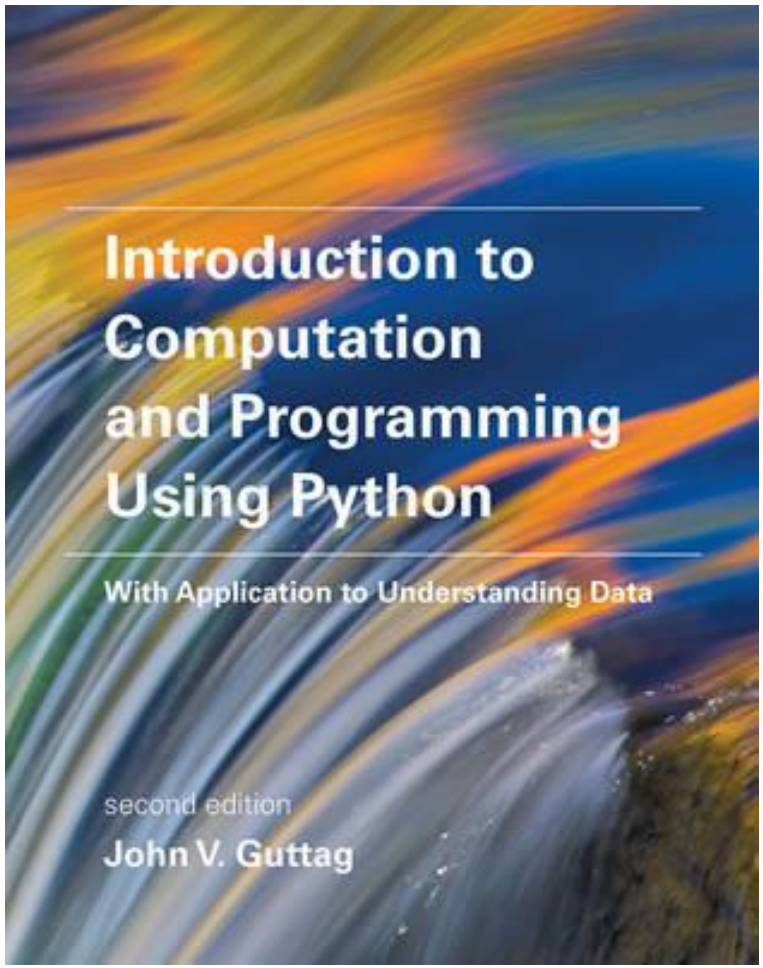
Introducción

Objetivo del curso

Que los participantes aprendan sobre la teoría básica de las ciencias de la computación, acompañada de su implementación en un lenguaje de programación de alto nivel.

Metodología

- Presentaciones y ejercicios en clase con los conceptos del libro *Introduction to Computation and Programming Using Python with Application to Understanding Data*.



Actividad	Punteo
Tareas y ejercicios	50 %
Proyecto 0	25 %
Proyecto 1	25 %

Recomendaciones

- Se debe buscar aprender la **lógica de programación** y no las recetas vistas en clase.
  - Lo importante es aprender a escribir programas para resolver problemas.
  - Esto te permite aprender a programar en otro lenguaje más rápidamente.
- ¿Eres totalmente nuevo en programación?
  - **PRACTICA, PRACTICA y ¡PRACTICA!**
  - No temas en probar instrucciones de Python, ¡la computadora no va a explotar!
  - **¡TEN PACIENCIA!**
- Puedes consultar un sinfín de cursos, blogs, tutoriales y documentación en Internet.

Agenda

- Introducción a la computación.
- Introducción a Python.

¿Qué hace una computadora?

- Fundamentalmente:
  - Realiza cálculos (hasta billones por segundo).
  - Guarda los resultados (cientos de GBs).
- ¿Qué tipo de cálculos?
  - Muy simples: operaciones aritméticas y lógicas.
  - Los que el programador **define**.

Tipos de conocimiento

- **Declarativo**: afirmación o enunciado a partir de un hecho.

  La raíz cuadrada de x es un número y tal que $y \times y = x$.

- **Imperativo**: una receta de cómo hacerlo.
  - Empiece con una deducción g.
  - Si $g \times g$ está cerca de x, deténgase y g es la respuesta.
  - De lo contrario, su nueva deducción es el promedio de g y x/g.
  - Repita el proceso hasta estar cerca de x.

Algoritmos

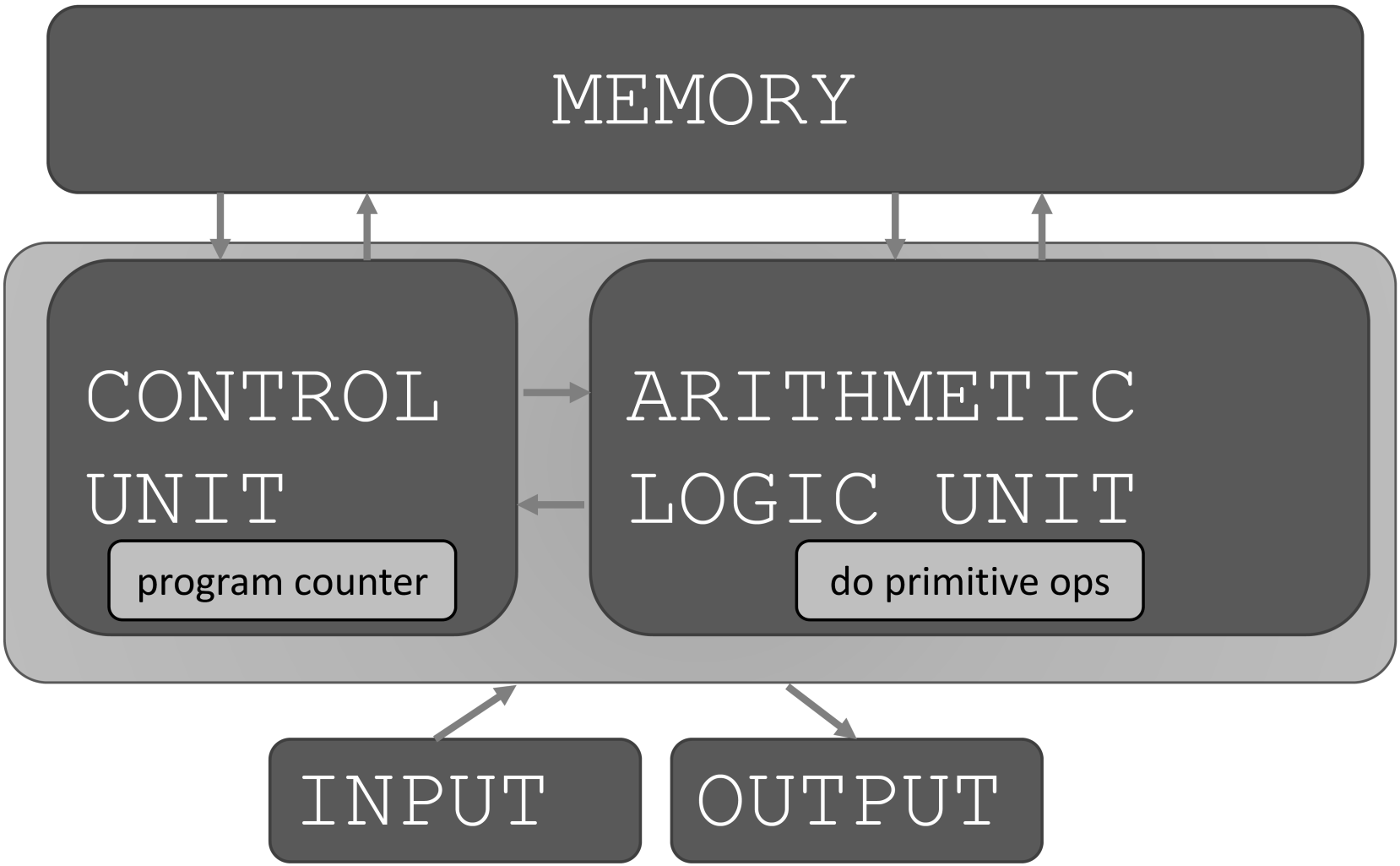
- Secuencias de pasos simples (*atómicos*).
- Hay un flujo de control que especifica cuándo se ejecuta cada paso.
- Existe una forma de determinar cuando parar.

Estos ingredientes conforman un algoritmo.

Tipos de computadoras

- ¿Qué exactamente es una computadora?
- Las computadoras son máquinas que capturan la receta.
- de programas fijos (e.g. calculadora).
- de programas almacenados: guardan y ejecutan instrucciones almacenadas en memoria.

Arquitectura básica



Programas almacenados

- Consisten en una secuencia de instrucciones almacenadas en la computadora.
- Construidas a partir de instrucciones primitivas:
  - Aritméticas y lógicas.
  - Condiciones simples.
  - Operaciones de memoria.
- El **intérprete** se encarga de ejecutar las instrucciones en orden.
- Utiliza pruebas para cambiar el flujo de control en la secuencia.
- Se detiene cuando termina la ejecución.

Máquina universal de Turing

"(...) In 1936, the British mathematician Alan Turing described a hypothetical computing device that has come to be called a Universal Turing Machine. The machine had an unbounded memory in the form of 'tape' on which one could write zeroes and ones, and some very simple primitive instructions for moving, reading, and writing to the tape. The Church-Turing thesis states that if a function is computable, a Turing Machine can be programmed to compute it."

Máquina universal de Turing

"(...) The Church-Turing thesis leads directly to the notion of Turing completeness. A programming language is said to be Turing complete if it can be used to simulate a universal Turing Machine. All modern programming languages are Turing complete. As a consequence, anything that can be programmed in one programming language (e.g., Python) can be programmed in any other programming language (e.g., Java). Of course, some things may be easier to program in a particular language, but all languages are fundamentally equal with respect to computational power."

Primitivas básicas

- Turing mostró que se puede computar **cualquier cosa** con algunas primitivas básicas.
- Los lenguajes de programación modernos tienen primitivas más convenientes.
    - Análogamente a un lenguaje natural, tienen palabras, sintaxis, semántica y significado.

Cualquier tarea computable en un lenguaje, es computable en cualquier otro lenguaje de programación.

Construyendo expresiones

- Las expresiones son combinaciones (complejas) legales de primitivas en un lenguaje de programación (LP).
- Las expresiones y cómputos tienen valores y significados en el lenguaje.
- Construcciones primitivas:
  - Español: palabras.
  - LP: números, cadenas de texto, operadores.

Sintaxis

- Se refiere a las reglas que especifican secuencias correctas de símbolos que pueden ser usadas para formar expresiones en el lenguaje.
- Por ejemplo: “Niño gato perro” no es sintácticamente válida en español.
- En Python, una secuencia correcta es `1.5+1.6`, pero no `1.5 1.6`.

Semántica estática

- Se refiere a que las cadenas sintácticamente válidas tengan significado.
- “Yo estamos hambriento” no es semánticamente válida.
- En Python:
  - `3.2*5` es válida.
  - `3+"hola"` no es válida.

Semántica

- Se refiere al significado asociado a las cadenas sintácticamente válidas y sin errores semánticos estáticos.
- “Estudiar el PES es alegre” puede tener diversos significados.
- Los lenguajes de programación están diseñados para tener solo un significado.
    - Aunque en ocasiones, este puede no ser el deseado por el programador.

Tipos de errores

- **de sintaxis**:
  - Muy comunes y fácilmente detectables.
- **de semántica estática**:
  - Algunos programan revisan antes de permitir la ejecución.
  - Pueden causar comportamiento impredecible.
- Sin errores semánticos pero **diferente significado al deseado**:
  - Los programas se detienen, fallan o se ejecutan para siempre.
  - La respuesta puede no ser la esperada.

Los programas deben escribirse de tal forma que, si fallan, el hecho de que no funcionen sea evidente.

¿Python?

Rank	Language	Type	Score
1	Python		100.0
2	Java		96.3
3	C		94.4
4	C++		87.5
5	R		81.5
6	JavaScript		79.4
7	C#		74.5
8	Matlab		70.6
9	Swift		69.1
10	Go		68.0

- Es un lenguaje interpretado de código abierto, relativamente simple, y excelente opción para principiantes.
- Es de propósito general:
  - Aplicaciones de escritorio y web.
  - Interacción con *hardware*, videojuegos.
  - Computación científica (*data science*, AI).
- Utilizado en la industria, academia y por muchos profesionales:
  - Google, Amazon, Dropbox, Instagram, Mr. Robot.

Python es interpretado

- Los lenguajes compilados convierten el código fuente a código de máquina.
- En los lenguajes interpretados, el intérprete convierte la fuente a su estructura interna y ejecuta línea por línea.

Programas de Python

- Un programa es una secuencia de definiciones e instrucciones.
  - Las definiciones son evaluadas: `a=2`, `x,y = y,x`.
  - Las instrucciones son ejecutadas por el intérprete: `print("Hola amigo")`.
- Las instrucciones indican al intérprete qué hacer.
- Pueden ser dadas directamente en la línea de comandos o en un archivo de texto.
  - Jupyter notebooks.

Objetos

- Los programas de Python manipulan **objetos**.
- Cada objeto tiene un tipo asociado, que define lo que se puede hacer con él.
- Los objetos pueden ser de tipo:
  - **escalar**: indivisibles (atómicos).
  - **no escalares**: poseen estructura interna.

Objetos escalares

En Python, existen 4 tipos de objetos escalares

- `int` — representa número enteros: 2, 3 - 2.
- `float` — representa números reales: `3.14159`
- `bool` — representa valores booleanos: `True` y `False`[1].
- `NoneType` — toma el valor especial `None`.
- Conversión de tipos con las funciones `int()`, `float()`, `bool()`.

Podemos utilizar la función `type()` para ver el tipo de un objeto.

[1] Python es *case sensitive*.

Impresión en consola

- Para mostrar salida de un código al usuario, utilizamos el comando `print`. Utilizado en archivos de código o *scripts*.

```
print("Hola amigos")
```

- Cuando estamos en la consola, basta con escribir el nombre de la variable o expresión para mostrar un resultado.

```
a=2
b=3
# Imprimir el resultado
a+b
5
```

Expresiones

- Se **combinan objetos y operadores** para formar expresiones.
- Una expresión tiene un **valor**, el cual tiene un tipo.
- Sintaxis: `<objeto> <operador> <objeto>`

Operadores

- Aritméticos: `+ - * / // % **`
- Comparación: `== < > <= >=`
- Lógicos: `and`, `or` y `not`
- Asignación (une nombre a valor): =

Notar la diferencia entre = y ==.

Considerar la precedencia de los operadores:

- Paréntesis
- `**`
- `*`
- `/`
- `+` y `-`

Ejecutados de izquierda a derecha, en orden de aparición.

Expresiones más abstractas

- Nombres de variables ⇒ **reutilizar** nombres y valores.

```
pi = 3.14159
radius = 2.2
area = pi*(radius**2)
```

- Se vale referir a la misma variable:

```
radius = radius + 1
```

- El valor de `area` no cambia hasta computarla nuevamente.