Lo básico del curso

Ayudantía 0

Jaime Castro Patricio López

Departmento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

IIC2233, 2015-1

Tabla de contenidos

Python

¿Qué sabemos?

- Es *Open Source*, es decir, su código es público y cualquiera puede verlo.
- Es Agnóstico de Sistema Operativo *(OS)*, o sea, no importa si es Windows, UNIX, etc.
- Es un lenguaje interpretado.
  - Requiere de un intérprete, eso es lo que instalaron cuando les pedimos que bajaran Python.
  - Al contrario de otros lenguajes que son compilados.
  - Los intérpretes están escritos en otro lenguaje, distinto a Python.

PEP8

Standard

"Code is read much more often than it is written."

— Guido van Rossum

PEP8

Guía de estilo para mantener la consistencia y leíbilidad.
https://www.python.org/dev/peps/pep-0008/

Ejemplo

¿Espacios vs Tabs?
Se deben usar 4 espacios para identar.

Sobre el curso

Repositorio del curso

- https://github.com/IIC2233-2015-1/syllabus
  - Se subirá todo el material de cátedra y ayudantía.
  - Está el programa del curso.
  - Hay una wiki con información.
  - Se utilizará como foro para preguntas.

Tarea para la casa (o ahora mismo)

Crear una cuenta Github para poder preguntar en el foro.

Sobre el curso

Material del semestre pasado

- Este semestre comenzó a hacerce el curso en Python,
- Todo material viejo les sirve conceptualmente.
- El curso es "Programación Avanzada", independiente del lenguaje. Lo importante es pensar como programador y usar Google para cualquier duda sobre el lenguaje.

Introducción a Git

¿Para qué?

- ¿Cómo revertimos un cambio en el código?
- ¿Cómo desarrollo una nueva funcionalidad sin arruinar las demás?
- Si mucha gente trabaja en un mismo código:
  - ¿Cómo evitamos colisiones en el código?
  - ¿Cómo sabemos qué fue modificado y quién lo hizo?

Introducción a Git

¡Habemus Git!

Definición

Sistema de control de versiones distribuido y Open Source que permite flujos de trabajo no-lineales.

Introducción a Git

Proveedores

Algunos proveedores de este servicio.

- Github, el más conocido y el que probablemente usemos.
- Bitbucket, otra gran alternativa.
- Puedes armar tu propio servidor git.

Uso

Glosario I

- Branch o rama:
  - Todo repo tiene una rama principal, por lo general: master.
  - Uno crea ramas a partir del estado actual de otra rama.
  - Uno puede ir cambiando entre ramas a gusto.
  - Una vez implementada la nueva funcionalidad, se hace un merge.
    - Se "fusiona" una sub-rama con su rama padre.
    - Ojo! A veces pueden surgir conflictos.
- Pull: bajar un estado del repositorio a la máquina local.
- Commit: guardar los últimos cambios en la rama actual.
  - Es como una foto que se le toma al código.
  - Guarda los cambios localmente en tu máquina.

Uso

Glosario II

- Stage: prepara los archivos para un *commit*.
- Clone: clonar un repositorio a una máquina local.
- Push: Subir los *commits* al repositorio.
- Stash: Es como un commit temporal para guardar los cambios.
- Fork: Crear un repositorio en base a otro a partir de cierto *commit*.

¿Dónde aprender?

- Ejemplo en ayudantía.
- Recursos para estudio personal:
  - https://try.github.io
  - https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

Módulos

Motivación

- En Introducción a la Programación se suele escribir todo el código en un mismo archivo. Esto no es viable cuando nuestros programas empiezan a crecer.
  - El mantenimiento del código es difícil
  - Es casi imposible trabajar en equipo
  - Se ve horrible
- Queremos tener una forma de poder dividir nuestro programa en varios archivos: módulos

Módulos

Definición

Definición

Un módulo es un archivo que contiene definiciones y declaraciones de Python. El nombre del archivo es el nombre del módulo.

- Básicamente cualquier archivo escrito en Python constituye un módulo.
- Python por defecto ya trae varios implementados

Ejemplo

Un archivo llamado fibonacci.py constituye un módulo llamado fibonacci

Módulos

Más sobre módulos

Ejercicio

Descargar el archivo `modulo_ayudantia.py` desde repositorio GitHub del curso.

- ¿Qué puede contener un módulo?
  - Variables
  - Métodos
  - Clases
- ¿Qué puedo hacer con un módulo?
  - Usarlo en otro programa o módulo

Import

Como usar los módulos

Para importar un módulo en nuestro programa, escribimos en el principio de él:

```
1  # Sentencias import
2  import nombre_modulo
3  # ...
```

Luego para usar algo del módulo importado:

```
1  # ...
2  nuevo_objeto = nombre_modulo.ClaseContenida()
3  variable = nombre_modulo.variable_contenida
4  nombre_modulo.funcion_contenida()
```

- El nombre del módulo importado se agrega al namespace del script

Import

Como usar los módulos

Ejercicio

Abrir el intérprete de línea de comandos de Python en la carpeta donde se descargó `modulo_ayudantia.py` e importarlo. Luego pruebe a usar sus funciones y clases.

Import

Otra forma de importar módulos

Existe otra forma de importar un módulo:

```
1  # Sentencias import
2  from nombre_modulo import funcion_contenida, ClaseContenida, variable_contenida
3  # ...
```

Para ocupar lo que importamos:

```
1  # ...
2  nuevo_objeto = ClaseContenida()
3  variable = variable_contenida
4  funcion_contenida()
```

- El nombre del módulo importado no queda en el namespace del script, sino que quedan las cosas que queremos importar directamente.
- Debemos colocar explícitamente todo lo que queremos traer del otro módulo

Import

Otra forma de importar módulos

Ejercicio

Abrir el intérprete de línea de comandos de Python en la carpeta donde se descargó `modulo_ayudantia.py` e importarlo, **pero ahora de esta forma**. Luego pruebe a usar sus funciones y clases.

Ejercicio

Crear un archivo llamado `math.py` con un único método llamado `hola_mundo()`, que deberá imprimir en pantalla "Hola Mundo". Luego abra el intérprete de Python en la carpeta donde guardó el módulo, impórtelo y llame a la función que creó.

- ¿Funcionó?

Import

Remarks

Ojo: Python buscará el módulo en el siguiente orden:

- Módulo integrado con Python
- Archivo nombre_modulo.py ubicado en la misma carpeta del archivo del que se importa
- Archivo nombre_modulo.py en el directorio de instalación.

Import

Haciendo trampa

Podemos incorporar al namespace todo lo que contiene otro módulo sin tener que nombrar las cosas de a una:

```
1  from nombre_modulo import *
```

Para usar el contenido de ese módulo se hace igual que antes.

Advertencia

Esto se considera en extremo una mala práctica.

Import

Main de un módulo

- Cuando importamos un módulo, se ejecuta todo lo que no esté declarado bajo una función o clase.
- Queremos que esto suceda sólo si lo llamamos directamente (`python nombre_modulo.py`)
- Solución: Agregar un condicional al módulo que contiene todo el código ejecutable

```
1 if __name__ == "__main__":
2     # Todo lo que quiera ejecutar si se llama al modulo.py
```

Final Remarks

- Usaremos GIT para entregar las tareas. Detalles: Próximamente
- No escriban todo su código en un único archivo. Usen los módulos
- Jamás ocupen `from modulo import *`. Se irán al infierno si lo hacen.
- Revisen como ocupar el respositorio del curso y como plantear preguntas.
- Bonus: Si le envían un correo a un ayudante o a los profesores, asegúrese de que el asunto cumpla con el patrón `IIC2233 - Mi Asunto`. Esto es para poder clasificar nuestros mensajes. Pero el foro es de preferencia, pues tu pregunta puede ayudar a los demás.

Lecturas y materiales I

- Última versión de Git
  http://git-scm.com/.
  2015-03-06.
- Try-Git, por CodeSchool y Github.
  https://try.github.io.
- PEP 8 - Style Guide for Python Code
  https://www.python.org/dev/peps/pep-0008/.
  01-Aug-2013.