PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
2^{do} semestre 2016

Guía Repaso Examen

1. Pa' pollos

1. **Testing**
   Un ambiente de pruebas para realizar testing, consiste en un conjunto de variables necesarias para los tests, que deben ser re-inicializadas antes de cada test y eliminadas al final de éstos. Explique cómo se genera un ambiente de prueba en `unittest`. Escriba un ejemplo en Python.

2. **Strings, GUI**
   Suponga que un servicio de correo electrónico recibe solo el e–mail (el browser ya llena automáticamente el password, despreocúpese de ese *textbox*) y al intentar acceder al servicio este verifica que es un e–mail válido. Una dirección de correo electrónico es válida si cumple con el formato: **x** @ **y**

   - Solo existe el caracter @ entre los strings **x** e **y**
   - El string **y** puede contener más de un dominio (ej: puc.*cl* , ing.puc.*cl*)
   - Solo se permiten dominios chilenos, *i.e*, dominio final **cl**
   - Ningún string puede ser vacío

   a) (**Strings**) Escriba en Python la función verificadora.
   b) (**GUI**) Dibuje y conecte los *widgets* principales de la interfaz con la función verificadora del *input* del usuario (preocúpese de manejar solo los elementos gráficos relacionados a esta función).

3. **Clases Abstractas**
   Explique el concepto de "Clase Abstracta", señalando cuáles son las principales ventajas que nos ofrecen este tipo de clases desde el punto de vista del modelamiento.

4. **OOP**
   Muestre un ejemplo que ilustre la diferencia entre herencia y composición en modelamiento OOP.

5. **Bytes**
   Explique qué ocurre, línea por línea, en el siguiente código y en cada print indique qué se imprime. Si no sabe la transformación número - letra, puede asumir una cualquiera.

```
a = bytearray(b"Hola")
print(a)
print(a[0])
a[0] = 104
```

```
print(a)
b = b"Hola"
print(b)
print(b[0])
b[0] = b"h"
print(b)
```

6. **Funcional**
   ¿Qué se imprime exactamente en el `print` al final del siguiente código?

```
f1 = lambda x: (x+1) ** 2
v1 = [[1, 2, 4], [1, 3], [2, 5, 9], [1, 6]]
v2 = list(map(lambda y: list(filter(lambda x: x > 10, list(map(f1, y))
    )), v1))
print(v2)
```

7. **Metaclases**
   Dado el siguiente código, responda las preguntas a continuación y justifique sus respuestas:

```
class Examen:

    def __new__(cls, *args, **kwargs):
        cls.students_dict = {}
        cls.id_ = cls.generate_user_id()
        return super().__new__(cls)

    def __init__(self, name):
        self.name = name

    def __call__(self, *args, **kwargs):
        return [Examen.students_dict[ar] for ar in args]

    @staticmethod
    def generate_user_id():
        count = 0
        while True:
            yield count
            count += 1

    def add_user(self, name):
        Examen.students_dict[name] = next(Examen.id_)

if __name__ == "__main__":
    e = Examen("Progra")
    e.add_user("E1")
    e.add_user("E2")
    e.add_user("E3")
    print(e.students_dict)
    print(e("E1", "E2", "E3"))
```

   a) ¿Qué representa `cls`?

b) El método `generate_user_id` ¿pertenece a la clase o a las instancias de `Examen`?

c) ¿Qué imprime la sentencia `print(e.students_dict)`?

d) ¿Qué imprime la sentencia `print(e(`“E1”, “E2”, “E3”`))`?

2. Pa' Mortales

1. **Serialización**

   a) Explique cómo se puede interceptar la carga de datos json (*json.load*) de tal forma de modificar el formato y/o contenido de los datos que se cargaron. Escriba un código como ejemplo.

   b) Explique cómo se puede personalizar la serialización de datos en formato json (*json.dumps*). Muestre un código de ejemplo.

2. **Regex**
   Suponga que un servicio de correo electrónico recibe solo el e–mail (el browser ya llena automáticamente el password, despreocúpese de ese *textbox*) y al intentar acceder al servicio este verifica que es un e–mail válido. Una dirección de correo electrónico es válida si cumple con el formato: **x** @ **y**

   - Solo existe el caracter @ entre los strings **x** e **y**
   - El string **y** puede contener más de un dominio (ej: puc.*cl* , ing.puc.*cl*)
   - Solo se permiten dominios chilenos, *i.e*, dominio final **cl**
   - Ningún string puede ser vacío

   Escriba en Python la función verificadora usando expresiones regulares

3. **Funcional**
   ¿Qué se imprime en el `print` al final del siguiente código?

```
def f2(b, item):
    lc = b[:]
    lc.remove(item)
    return lc

def f1(a):
    if len(a) == 0:
        return [[]]
    return [[x] + y for x in a for y in f1(f2(a, x))]

print(f1(["a","b","c"]))
```

4. **Funcional**
   Escriba una función recursiva que aplique una función cualquiera recibida como argumento a una lista anidada que contiene listas de números enteros, por ejemplo:

```
print(tu_func(lambda x: x*x, [1, 2, [3, 4, [3,5]]]))
out:  [1, 4, [9, 16, [9, 25]]]
```

3. Pa' coreanos

1. **Networking, Grafos**

   a) Se requiere elaborar un sistema de predicción de temperatura para las distintas zonas de en una ciudad. En cada zona existe una plataforma capaz de predecir el tiempo hasta 10 días en el futuro. Esta plataforma requiere de la siguiente información para hacer la predicción: la temperatura actual en su zona local, medida a través de un sensor, y la temperatura actual en las zonas vecinas. Existe un servidor que envía a cada zona una lista de tuplas serializadas mediante `pickle`, donde cada tupla incluye el host y el puerto de cada equipo de medición en las zonas vecinas a cada nodo, de tal forma que puedan conectarse e intercambiar la información necesaria. Cada vez que la temperatura en alguna zona cambia, debe realizarse una nueva predicción para el día en cuestión y los diez días subsiguientes de cada zona afectada. Cada zona no puede acceder en forma directa al valor de los sensores de sus vecinos, pero si al valor del sensor que está dentro de la misma zona.

   Suponga que existe una función que recibe como argumentos la información de temperatura de su zona y la de los vecinos, y retorna la temperatura para los próximos 10 días en la zona.

   Implemente la solución correspondiente que trabajará en cada una de las estaciones de medida de la ciudad. Asuma que el servidor que retorna la lista con las zonas vecinas ya existe y usted sólo debe preocuparse de conectarse a él. También suponga que la función que realiza la predicción está implementada y que existe una función que retorna el valor del sensor para cada zona.

   b) Considerando el mismo contexto de la pregunta anterior, suponga que usted debe implementar el servidor que se encarga de enviar a cada nodo la lista con la información necesaria de sus nodos vecinos. El servidor debe recibir como input un nodo del grafo que representa las conexiones entre las zonas (ejemplo, ver figura 1). Considere que el nodo contiene: un host, un puerto y una lista con los nodos vecinos en el grafo.

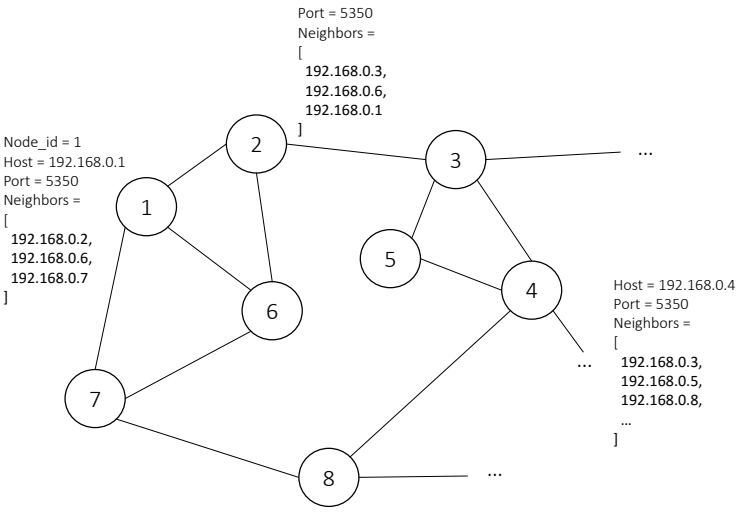

Figura 1: Ejemplo del grafo recibido como input por el servidor.

2. **Simulación, Threads, OOP**

   a) **Modelación OOP**
   En un curso de desarrollo de software se arman equipos de `k` alumnos que trabajan para un cliente. Cada grupo de alumnos es supervisado periódicamente por un ayudante y el profesor del curso.

Cada alumno tiene una probabilidad de encontrarse enojado y una probabilidad de renunciar. Además, un alumno puede programar un cierto número de líneas de código por cada tarea que se le encomienda, que depende del lenguaje:

- **Python:** Random entre 1 y 600 líneas
- **Javascript:** Random entre 1 y 400 líneas
- **C:** Random entre 1 y 200 líneas

Dentro del equipo hay un alumno especial que nunca renuncia, el jefe de grupo. Tanto el profesor como el ayudante pueden realizar comentarios. Cada uno de ellos tiene una probabilidad de decir un comentario positivo (y en caso contrario, un comentario negativo). Cada ayudante tiene una probabilidad de solucionar un conflicto del grupo que supervisa. Mientras tanto, el cliente tiene una probabilidad de máximo un 10 % de hacer *boicot* al proyecto, ya sea porque el resultado no le gustó, o porque es tan bueno que no lo puede entender. Modele esta situación con OOP implementando las clases necesarias en Python.

b) **Threads, Simulación**
En el mismo contexto anterior, cada semana hay una reunión del equipo junto al profesor y el ayudante. En esta oportunidad, el ayudante y el profesor emiten un comentario, que puede ser contradictorio (uno positivo y el otro negativo). Durante el desarrollo de la tarea, hay un conflicto si es que dos o más de los integrantes involucrados están enojados con probabilidad p. El ayudante puede solucionar con probabilidad q el conflicto del equipo por reunión. Existe una reunión mensual del equipo junto al cliente, donde el cliente podría *boicotear* el trabajo realizado.
Cada cierto tiempo aleatorio (entre 0 y 7 días), parte del equipo se junta para realizar una tarea. Al inicio de ésta, se elige un lenguaje al azar entre `Python`, `Javascript` o `C` para trabajar. Las líneas de código escritas corresponden a la suma de lo que cada integrante puede escribir en ese lenguaje. Además, es posible que uno o más integrantes decidan renunciar.

Usted debe generar 20 simulaciones en paralelo con threads, para 3 meses y un grupo de 9 personas. Ocupe las clases implementadas de la parte a). Para cada simulación, debe extraer el número de veces que el ayudante se contradijo con el profesor, el número de tareas realizadas por el equipo, el número de líneas de código escritas en total, la cantidad de conflictos que ocurrieron, la cantidad de conflictos solucionados, el número de integrantes que renunció y el número de veces que el cliente intentó *boicotear* el proyecto. **HINT:** `rand()` $< p$ retorna `True` con probabilidad p.

3. **Decoradores**
Implemente un decorador que controle el acceso a la función decorada de tal forma de que no pueda volver a ser ejecutada mientras ya esté siendo ejecutada.

4. **Threads, Simulacion, Grafos**
Se desea evaluar un proyecto de infraestructura vial, donde se propone la construcción de un sistema de carreteras que supuestamente va a tener un impacto importante en el flujo del tráfico de la ciudad de Santiago. El estado actual de la ciudad está descrito a través de un grafo no dirigido (todos los caminos son bidireccionales), donde los nodos corresponden a ciertos sectores críticos predefinidos de la ciudad. Dos nodos están conectados en el grafo si existe una secuencia de calles bidireccionales que conectan los dos sectores de la ciudad representados por los nodos. El tiempo estimado de viaje entre dos nodos depende de: la distancia, el número de semáforos y la afluencia de tráfico. Existe una función f_v que dada una hora y día de la semana retorna el número estimado de vehículos en la ciudad.
Para simular la existencia de semáforos y calles, existe una función f_s que dados dos nodos i y j, retorna dos listas: una con los tiempos entre cambios de colores (rojo-verde) para cada uno de los semáforos entre los nodos i y j, y otra con los tiempos medios (μ's minutos) de viaje entre semáforos (cada tiempo distribuye expovariate($\frac{1}{\mu}$)). Si los nodos i y j no se pueden conectar, la función genera una excepción (debe evitar que se genere).

Cuando la luz de un semáforo es roja, los vehículos forman una cola para cruzar, esto produce que un vehículo además de esperar que cambie la luz debe esperar que todos los que lo anteceden hayan cruzado. Este tiempo extra de espera distribuye uniforme $[0, \frac{c}{2}]$ minutos, donde c es el número de vehículos que lo anteceden en la cola.

El grafo está representado por una matriz donde en la posición i, j hay un 1 si los nodos i, j están conectados, 0 en caso contrario (imagine una línea recta con varios semáforos entre dos nodos).

Implemente, usando **Threads**, el código en Python que genere una simulación de un día de circulación de vehículos en la región. La cantidad de vehículos inicial es 0 (la función f_v debe ser llamada en $t = 0$). Cada vez que la función f_v retorne un número de vehículos mayor al número actual, usted debería generar nuevos vehículos que comiencen a circular, el punto de partida debe ser un nodo seleccionado aleatoriamente. Si la función f_v retorna un número de vehículos menor al que está circulando actualmente, usted deberá retirar vehículos. Para retirar un vehículo usted debe interrumpir su trayectoria, obligándolo a que viaje de vuelta al nodo de partida, una vez que llega al nodo, puede eliminarlo de la circulación. Cada trayectoria debe ser generada aleatoriamente.

Usted deberá consultar a la función f_v cada una hora, además **debe generar un registro de la cantidad de viajes terminados**, es decir, cuántos autos llegan a su destino original sin tener que devolverse al origen. Deje explicitados todos los supuestos que utilice, solo serán válidos mientras no violen el enunciado