Trabajo Práctico 2 Python/Programación
Matemática III Econ., turno noche, 2019 cuat. 1.
IDEI-UNGS.

La parte más importante de enseñar es enseñar qué es saber.

Simone Weil (1909-1943)

Autor: G. Sebastián Pedersen — sebasped@gmail.com — Vie 21-Jun-2019.

1. Códigos ejemplo

1.1. Calculando una suma con un ciclo

El siguiente programa en Python calcula la suma:

$$\frac{1}{x^1} + \frac{1}{x^2} + \cdots + \frac{1}{x^{n-1}}$$

para un valor de n antes fijado. Define una función `suma`, que luego utiliza para $x = 3$, y finalmente imprime el resultado. A continuación el código:

```
# -*- coding: utf-8 -*-
n=10

def suma(x):
    i=1
    total = 0
    while i < n :
        total = total + 1.0/x**i
        i = i + 1
    return total

resultado=suma(3)
print(resultado)
```

cicloSuma.py

Observar el detalle del `1.0` en la división para evitar que Python realice una división entera.

1.2. Encontrando una raíz por Newton-Raphson

El siguiente programa en Python calcula (aproximadamente) la única raíz de la función $f(x) = -3x - 2^x$, es decir resuelve (aproximadamente) la ecuación

$$-3x - 2^x = 0$$

utilizando el método de Newton-Raphson[1]. Los parámetros a y b están definidos previamente a la función `f`, luego llama a la función e imprime el resultado. Finalmente comprueba que el resultado es efectivamente una raíz. A continuación el código:

[1] `https://es.wikipedia.org/wiki/M%C3%A9todo_de_Newton`

```python
# -*- coding: utf-8 -*-
from scipy import optimize

a=-3
b=2

def f(x):
    func = a*x - b**x
    return func

raiz = optimize.newton(f, 0.5)
print(raiz)

comprobar = a*raiz-b**raiz
print(comprobar)
```

ejemploNR.py

2. Enunciado del TP

I) Para un caso sencillo ($n = 4$ y $x = 2$ por ejemplo) describir en detalle todas las iteraciones del `while` de la sección 1.1, indicando en cada iteración los valores de `i`, `n`, `x` y `total`, y por qué el ciclo se sigue ejecutando o por qué para.

II) Explicar la diferencia entre definir una función y utilizar una función ya definida. Se puede explicar a partir de los códigos ejemplo de las secciones 1.1 y 1.2.

III) Elegir alguna ecuación donde aparezca la Yield to Maturity (YTM) y la misma no se pueda despejar. Explicar la elección de la ecuación (de mínima el origen) y el significado de cada uno de sus parámetros.

IV) Hacer un programa en Python que resuelva (aproximadamente) por Newton-Raphson la anterior ecuación (es decir, calcule la YTM para valores de los parámetros ya fijados).

V) Hacer varias corridas del programa variando los valores de los parámetros, e interpretar los resultados. Se pueden elegir variaciones de los parámetros que se consideren relevantes (es decir, no simplemente variar al azar y correr el programa). No es necesario hacer algo exhaustivo.

VI) (Bonus) Explicar cómo se usa el valor inicial de Newton-Raphson (el `0.5` del código ejemplo de la sección 1.2). Explicar cómo funciona Newton-Raphson, tanto la idea del método como el algoritmo propiamente dicho, y dar ejemplos de cúando funciona bien y cuándo tiene problemas. Investigar métodos superadores.

3. Pautas del TP

a) Se deberá realizar individualmente o en grupo de 2 (dos) integrantes.

b) Se deberá entregar *únicamente* un informe en pdf respondiendo los ítems de la sección 2.

c) El ítem bonus de la sección 2 es opcional, y se puede responder parcialmente.

d) El informe podrá además contener todas las aclaraciones o materiales extras que se consideren relevantes.

e) La entrega deberá ser por correo electrónico a `sebasped@gmail.com` antes del Lun 24-Jun-2019 a las 23:59 hs. (hora local).

f) Se podrán realizar consultas por correo electrónico a `sebasped@gmail.com` hasta el Sáb 22-Jun-2019 a las 23:59 hs. (hora local).

g) Posterior a la entrega del TP, se podrá requerir a lxs integrantes del grupo (juntxs o por separadx), que expliquen o amplíen cualquier cosa en relación a la entrega.