UNA. Escuela de Informática
EIF400 Paradigmas de Programación II-2018
Dr. Carlos Loría-Sáenz

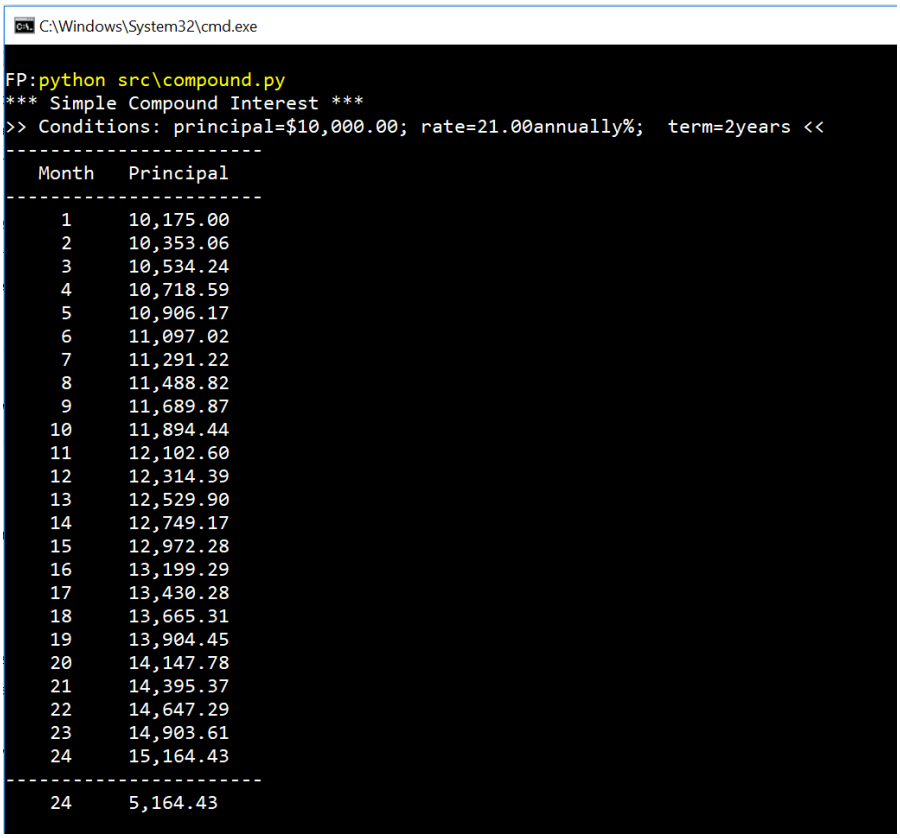
Práctica 03 `FP-OOP Python Básico`

Objetivos

Ejercitar de manera básica conceptos paradigmáticos de programación funcional (`FP`) y orientada a objetos (`OOP`), usando `Python` como caso de estudio.
Ver: directorio `src` adjunto

Tareas

1) Implementamos un algoritmo demo de cálculo de interés compuesto. Estudie las versiones imperativas que se le adjuntan. Re-implemente en estilo FP-Python eliminando todo estilo imperativo. Hágalo en dos fases: a) Una que usa listas como la versión entregada. b) Otra usando "pipes" que evita usar listas (memoria) intermedia.



2) El conocido método de Newton_Raphson permite estimar raíces (ceros) de funciones derivables generando aproximaciones sucesivas por medio de la recurrencia siguiente partiendo de un valor inicial r_0 (y asumiendo que la derivada no se hace cero en el vecindario de búsqueda inducido por el valor inicial):

$$r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)} \; n = 0,1,2 \ldots$$

Usando un estilo FP escriba una aplicación que encuentre raíces de polinomios por este método. Asuma, por simpleza, que modelamos un polinomio como un (sub)tipo de lista de coeficientes con el monomio de mayor grado a la izquierda y poniendo cero si ese monomio no aparece en el polinomio.

Por ejemplo $x^2 - 1$ sería como $[1, 0, -1]$. Y $x^3 - 7x^2 + 7x + 15$ sería como [1,-7, 7, 15].
Implemente la clase `Poly` adjunta usando FP en todo lo que sea apropiado. Deben desaparecer todos los "to do" indicados. Los parámetros `epsilon` y `max` controlan la precisión y máxima cantidad de veces a recurrir para encontrar una raíz.
Debe implementar un método que usando división sintética y Newton-Raphson trata de encontrar todas las raíces.
Salida actual sin implementaciones

```
FP:python src\newton_raphson.py

*** TO DO: MUST IMPLEMENT '__init__' ***
DETAIL # 0 --> Must be inmutable <--

DETAIL # 1 --> args must a list of numbers <--


*** TO DO: MUST IMPLEMENT '__str__' ***

*** TO DO: MUST IMPLEMENT '__repr__' ***
[1, -7, 7, 15]

*** TO DO: MUST IMPLEMENT '__call__' ***
3

*** TO DO: MUST IMPLEMENT 'dx' ***
DETAIL # 0 --> Must be property <--

DETAIL # 1 --> Calculated just once if called several times <--


*** TO DO: MUST IMPLEMENT '__str__' ***

*** TO DO: MUST IMPLEMENT '__repr__' ***
[1, -7, 7, 15]

*** TO DO: MUST IMPLEMENT 'solve' ***
DETAIL # 0 --> r0 must be a number <--


*** TO DO: MUST IMPLEMENT 'degree' ***
DETAIL # 0 --> Must be property <--

5.0

*** TO DO: MUST IMPLEMENT '__floordiv__' ***
DETAIL # 0 --> alpha must be a number <--

DETAIL # 1 --> Returns a new poly <--


*** TO DO: MUST IMPLEMENT '__str__' ***

*** TO DO: MUST IMPLEMENT '__repr__' ***
[1, -7, 7, 15]

*** TO DO: MUST IMPLEMENT 'roots' ***
DETAIL # 0 --> Must be a generator <--


*** TO DO: MUST IMPLEMENT 'solve' ***
DETAIL # 0 --> r0 must be a number <--


*** TO DO: MUST IMPLEMENT 'degree' ***
DETAIL # 0 --> Must be property <--

[5.0]

FP:
```