Programación en Python. Marzo 2020. IAI – EcyT - UNSAM.

En este "modo virtual" de clases vamos a pedirles que lean literatura específica para cada clase. Existe mucha literatura buena sobre Python. Nosotros los vamos a acompañar con ejercicios para ilustrar y reforzar los conceptos que allí se explican.

Vamos a mantener el régimen de ejercicios para hacer "en clase" que permiten demostrar o poner a prueba un concepto. La idea es pausar la lectura, implementar un programa pequeño que pruebe un concepto para asegurarse que lo entendieron, y seguir leyendo el próximo punto. Algunos de estos ejercicios están en el libro recomendado, otros están en la guía de actividades de cada clase. Además, y como antes, habrá ejercicios mas largos para resolver "en casa".

Lo mejor es entregar por mail (python@unsam.edu.ar) todos los ejercicios antes del inicio de la siguiente clase, y completar la autoevaluación. La discusión de estrategias y las consultas, aún más que antes son bienvenidas preferentemente al grupo whatsapp de la materia.
El libro recomendado es: http://docs.python.org.ar/tutorial/pdfs/TutorialPython3.pdf

Dicho libro, estas guías, el formulario de autoevaluación y otros documentos de la materia se encuentran en bit.ly/UnsamPython2020C1

En la Clase 1 llegamos a ver Listas, `if ... elif ... else:`, y ciclos con `while`. Es bueno leer esos mismos temas del libro, para saber como presenta el libro cosas ya conocidas por ustedes. En la página 9 (sección "**Cadenas de caracteres**") introducen lo que llamamos strings y en la 13 introducen Listas (sección "**Listas**"). Lean esas seciones para repasar y tengan a su lado una computadora donde probar lo ejemplos del libro y los nuestros. Esto último es importante. No se aprende a hablar un idioma con sólo leerlo.

Tambien es importante, para los que nunca usaron el debugger de Spyder, que sigan un pequeño tutorial de 6 páginas que se llama **"GuiaDebug.pdf"** y está en la carpeta "Clase 02" del Drive de la materia. Un debugger es algo que van a necesitar de aquí en más.

Clase 2

Aquí comienza la Clase 2: ciclos `for`, `print` con formato, `def` (definición de funciones) diseñada para acompañarlos a leer desde la sección "Más herramientas para control de flujo" del en la página 16 del libro TutorialPython3. Por favor léanla.

La sentencia `for ... in ...`

Prueben el primer ejemplo ("`#Midiendo cadenas de texto`") en sus Spyder.
Notar que `for p in palabras:` itera una vez por cada elemento de la lista `palabras`, es decir, itera sobre los elementos de una lista, donde cada elemento es una palabra.

Hay un error en el comentario del segundo ejemplo, notación `palabras[:]` debería decir:
`for p in palabras:` (en lugar de `"for w in words"`)

Porqué crearía una lista infinita ? (pensarlo antes de seguir…)
Porque cada vez que agrega un elemento a la lista `palabras` , la lista tiene un elemento más sobre el cual iterar, y por lo tanto la iteración nunca llega al último elemento y el ciclo nunca termina, creando una lista infinita.

Algo notable es que un `string` también es iterable. Probar en Spyder el Ejercicio 1 de la Guía de Actividades. En ese caso la unidad de iteración es una letra, y se itera para toda

letra en `palabra`. La instrucción `print (letra)` es invocada para todo valor que pueda tomar `letra`. Del mismo modo son iterables las listas, los arrays, y otras estructuras de datos que veremos mas adelante.

Es momento de hacer los Ejercicios 2 y 3 de la Guía.

Leer la función `range()` del libro. Esta función genera una secuencia iterable, lo cual es en sí muy útil, pero es especialmente adecuada para usarla en ciclos `for...in`. La función permite dar el valor inicial, el valor antes del cual detenerse, y el paso.

Es el momento preciso para hacer el Ejercicio 4

Leer las sentencias `break`, `else`, `continue` en lazos. (lazo = ciclo, bucle, loop)
Creo que esto requiere cierta explicación: Muchas veces es útil poder hacer excepciones dentro de un ciclo para casos determinados. No es común usarlas, pero en algunos casos pueden ser útiles. La instrucción `break` permite salir prematuramente de un ciclo `for` o un `while`. Si existe un `else:` éste sólo será ejecutado si no salimos con `break` antes, como se muestra en el buscador de números primos del libro. Dicho de otro modo: lo normal es ejecutar el `else:` al final del ciclo. Para evitarlo, usamos un `break`. Atentos a la indentación.

Si les dá hambre pueden hacer el Ejercicio 5

Pueden leer la sección "Definiendo Funciones"

La palabra reservada `def`

Aunque lo vimos la primera clase, aquí se trata específicamente la forma de definir funciones, recibir parámetros, controlar el flujo del programa en la función y definir valores de retorno. Prestar atención a los `docstrings` ("Cadenas de texto de documentación"), si se acostumbran a usarlos rutinariamente les ahorrará mucho tiempo y les permitirá colaborar con otra gente.

Aunque en esta sección se cubren varias formas de pasar parámetros a funciones, y varios tipos de parámetros, la forma del ejemplo 2 ("función que retorna una lista con los números de la serie de Fibonacci en lugar de imprimirlos") describe la gran mayoría de las funciones

que solemos utilizar: una función con `docstring`, con parámetros pasados por posición, que devuelve todo resultado con un `return`.

Sección “Más sobre listas”

Es importante detenerse en cada uno de estos métodos y evaluar su utilidad. Todos son muy poderosos y encierran tareas que son difíciles de programar eficientemente. Trataremos de ejercitarlos. La pequeña nota “Quizás hayas notado ...” que aparece debajo del ejemplo de las frutas es muy importante: muchos de estos métodos no devuelven un valor, pero de todos modos modifican la lista.

Vamos a leer hasta la sección “Usando listas como colas” (incluída), y dejar la comprensión de listas y las t-uplas para la próxima clase.

Con estas herramientas tratemos de resolver los demás ejercicios de la guía.

No olviden enviar ejercicios resueltos y preguntar lo que necesiten al grupo.