Banco de Preguntas Parametrizadas

Luis Ernesto Carrera Retana

9 de septiembre de 2020

1. Información general

Este proyecto consta de los siguientes programas:

- `generar` Esta es la función que genera las pruebas. Requiere dos archivos como argumentos: el primero es el archivo `ppp` donde se guarda la información general de la prueba, y el segundo la carpeta con las listas de los grupos de las personas estudiantes (cada uno de ellos descargado del tec digital y convertido a CSV) o un solo archivo csv. En esta misma carpeta guarda las evaluaciones.
- `evaluar` Esta es la función que realiza la evaluación. Requiere el archivo `ppp`, la carpeta con las listas de los estudiantes, y el archivo de las respuestas descargado de Microsoft Forms, y convertido a .csv, con ';' como separador entre columnas.
- `visualizar` Esta función recibe como argumento la dirección del archivo de una pregunta, y de manera opcional el número de ejemplos por generar, para generar un pdf de la pregunta. Esto permite revisar que la pregunta esté bien definida, y ver si genera los resultados esperados.

1.1. Requisitos

- python
- LaTeX(miktex o texlive)
- perl (no se requiere para la función `visualizar`)
- ghostscript (no se requiere para la función `visualizar`)

1.2. Modificando el LaTeX

El encabezado predeterminado se puede visualizar en el archivo `latex.py`.

2. Estructura del archivo general

```
<Escuelas>
__Lista de escuelas__
```

```
<Semestre>
__Semestre y año__

<Tiempo>
__Duración de la prueba__

<Cursos>
__Nombre de los cursos__

<Titulo>
__Título de la prueba__

<Encabezado>
__Paquetes, comandos nuevos, etc, para el encabezado del archivo LaTeX__

<Instrucciones>
__Instrucciones del examen__

<Seccion[, orden = aleatorio]>
  <Titulo>
  __Opcional si es sólo una sección__

  <Instrucciones>
  __Opcional si es sólo una sección__

  <Preguntas>
    [puntaje = __int__,] origen = __string__[, muestra = __int__]
<Fin>
```

- Tanto el nombre del curso como el título deben ser solamente **una** línea de texto debajo de la etiqueta respectiva.
- Las instrucciones pueden abarcar varios renglones e incluir líneas en blanco, para que LaTeX separe los párrafos.
- En las instrucciones *no* debe aparecer el símbolo de abrir etiquetas (ver `abrir` en `Info.py`) como primer carácter del párrafo, porque es la forma de determinar que allí finalizan.
- No debe haber espacios en blanco entre las especificaciones de las preguntas, pero sí se permiten comentarios.
- El `origen` de la pregunta puede ser un archivo con extensión `.tex` o una carpeta.
- Si la dirección es una carpeta, entonces el puntaje para cualquiera de las preguntas es el mismo, y se toma de acá. Si el puntaje no aparece, entonces el predeterminado es 1 punto.
- La muestra se refiere al número de preguntas que se toma de la carpeta. Si no aparece, el predeterminado es 1.
- Las preguntas finalizan con una línea en blanco.

3. Estructura del archivo para cada pregunta

```
<tipo = seleccion unica[, orden = aleatorio]>

<variables>
__nombre_de_variable__ = __expresion__

<pregunta>
__texto de la pregunta__

<item>
__respuesta o distractor__
```

1. Se asume que la respuesta correcta es el primero de los items. **Se debe dejar una línea en blanco al final de cada item**.

2. Las variable son opcionales.

4. Funciones

Se tienen dos tipos de funciones. Las funciones que únicamente se pueden llamar para definir variables, y las funciones generales que se pueden llamar en las variables, en la pregunta, y en los ítemes.

En las variables se define de manera normal. En la pregunta y los items cualquier expresión que requiera ser evaluada debe escribirse como una @-expresión:

- `@<__expr__>`
- `@{__expr__}`
- `@(__expr__)`
- `@[__expr__]`
- Una `@` seguida de cualquier símbolo, que es el mismo que se utiliza para cerrar, por ejemplo podría ser `@|__expr__|`.

¡El único requisito, es que el símbolo para cerrar **no** debe aparecer en la expresión a evaluar!

Bueno, no es el único. Toda a expresión a evaluar debe estar contenida en una sola línea.

Si el resultado de una @-expresión es un `string` o un entero, entonces se concatena al texto; si es un punto flotante, entonces se trabaja de manera predeterminada con 3 cifras significativas, y se imprime el número en notación decimal o en notación científica, dependiendo de cómo se permita saber de la forma más clara que se tienen 3 cifras significativas. Si el resultado de la @-expresión es otra cosa, entonces se deja que python lo convierta a texto, y se concatena.

4.1. Funciones para definir variables

Estas funciones únicamente están disponibles para definir variables:

4.1. `randrange(stop)`
`randrange(start, stop[, step])`

Un elemento `n` seleccionado al azar tal que `start<=n<stop`.

4.2. `randint(a,b)`

Un elemento `n` seleccionado al azar tal que `a<=n<=b`.

4.3. `choice(<seq>)`

Un elemento `n` seleccionado al azar de la sucesión no vacía `seq`.

4.4. `shuffle(<seq>)`

Reordena *in situ* la sucesión `seq`.

4.5. `sample(<list>, k)`

Toma una muestra de tamaño `k` de la lista `list`. `k` debe ser menor o igual al tamaño de la lista. La muestra no está ordenada con respecto a la lista.

4.2. ¿Cómo construir una sucesión?

Las sucesiones se definen mediante una de las siguientes formas:

- `range(<stop>)`
- `range(<start>, <stop>)`
- `range(<start>, <stop>, <step>)`

En el caso de un solo argumento, entonces la sucesión comienza en 0 y termina en `stop - 1`. Si tiene dos argumentos, entonces comienza en `start` y finaliza en `stop-1`. Con tres argumentos la función `range` define la sucesión `start`, `start + step`, `start + 2*step`, ..., `start + k*step`, donde `start + k*step < stop <= start + (k+1)*step`.

1. Si la sucesión está dada por los j elementos $0, 1, 2, \ldots, j-1$, se construye con `xs = range(j)`.
2. Si la sucesión está dada por los j elementos: $i, i+1, i+2, \ldots, i+j-1$, se construye con `xs = range(i, i+j)`.
3. Para una sucesión aritmética de k elementos $i, i+d, i+2d, \ldots, i+(k-1)d$, se construye con `xs = range(i, j, d)`, donde $i+(k-1)d < j \leq i+kd$.
4. Para concatenar dos o más sucesiones:
   `xs = [*range(<start>,<stop>[,<step>]), *range(<start>,<stop>[,<step>])]`.

Si lo que se quiere es tomar un elemento aleatorio de una sucesión simple, entonces mejor utilizar las funciones `randrange` o `randint`. Para el caso en que se concatenan dos o más sucesiones, entonces se puede utilizar `choice`.

4.3. Funciones generales

Lo que se tiene es un subconjunto de funciones de python, y algunas programadas específicamente para la generación de pruebas.

4.6. `+ - * / //`

Suma, resta, multiplicación, división normal (con punto flotante como respuesta) y división entera.

4.7. `range(stop)`

`range(start, stop[, step])`

4.8. `__exp_True__ if __exp_bool__ else __exp_False__`

Un `if` en un sólo renglón. En las expresiones booleanas se puede utilizar `not`, `or` y `and` según se necesite.

4.9. `round(__numero__[, __dig__])`

Redondea un número, y se puede especificar el número de dígitos a utilizar. 0 es el valor predeterminado.

4.10. `pow(a, b)` a^b

4.11. `abs(a)` $|a|$

4.12. `binomial(a,b)` $\binom{a}{b}$

4.13. `factorial(a)` $a!$

4.14. `gcd(a,b)` Máximo común divisor de a y b.

4.15. `sqrt(a)` $\sqrt{a}$ como punto flotante.

4.16. `factores(a)`

Factorización de a. Por ejemplo `factores(1000)` devuelve la lista `[(2, 3), (5, 3)]`, que representa a $2^3 \cdot 5^3$.

4.17. `txtFrac(a, b[, conSigno=False])`

Simplifica y escribe la fracción respectiva en LaTeXusando `dfrac`. `conSigno` es un booleano (`False` o `True` con `False` como predeterminado) que escribe un signo + antes de la fracción si es positiva.

4.18. `txtRaiz(a[, n=2[, conSigno=False]])`

Simplifica y escribe la raíz respectiva en LaTeX. `conSigno` es un booleano (`False` o `True` con `False` como predeterminado) que escribe un signo + antes de la raíz si es positiva.

4.19. `txtCoef(a[, conSigno=False])`

Coeficiente que precede a una variable. Si $a = 1$ entonces no se escribe (o se escribe solo un signo + si `conSigno`. De manera similar ocurre si $a = -1$. En caso contrario imprime el valor.

4.20. `txtExpo(n)`

Si `n == 1` no imprime nada. Si no imprime `^{n}`.