

# Índice general

<b>I</b>	<b>Dando formato a las preguntas</b>	<b>2</b>
1.	Estructura del módulo <code>qbank._export</code>	3
2.	Preguntas en $\text{\LaTeX}$ para usar con <b>Auto-Multiple-Choice</b>	4
2.1.	Escritura de las cuestiones dentro del entorno <code>choices</code>	5
2.1.1.	Texto de las cuestiones en función de si son verdaderas o falsas	5
2.1.2.	Última opción por defecto ( <code>lastchoices</code> )	5
2.2.	Información sobre las cuestiones descartadas (para revisión del profesor)	6
3.	Escritura del código $\text{\LaTeX}$ para las preguntas en formato <b>AMC</b>	7
3.1.	Preguntas de opción múltiple en formato para exámenes	7
3.1.1.	Función <code>AMC</code>	7
3.1.2.	Función <code>AMClastCh</code>	7
3.1.3.	Función <code>AMCmc</code>	8
3.1.4.	Función <code>AMClastChmc</code>	8
3.2.	Preguntas de opción múltiple en formato para revisión	8
3.2.1.	Funciones <code>AMCProfe</code> y <code>AMCmcProfe</code>	9
3.3.	Ejemplo de uso	9
3.4.	Variante para preguntas Verdadero/Falso en formato para exámenes en el aula	11
3.4.1.	Ejemplo de uso	11
4.	Preguntas en <b>XML</b> para usar con <b>Moodle</b>	12
4.1.	Preámbulo de los ficheros $\text{\LaTeX}$ para exportación a Moodle	12
4.1.1.	Codificación de caracteres no ASCII	12
4.2.	Estructura de preguntas de opción múltiple con el paquete <code>moodle</code>	13
4.2.1.	Parámetros opcionales del comando <code>\item</code>	13
4.2.2.	Parámetros opcionales del entorno <code>multi</code>	13
4.3.	Escritura automatizada de las cuestiones desde Python	14
4.3.1.	Procesamiento de las variantes extraídas del iterador	14
4.3.2.	Gestión de la opción por defecto ("Ninguna de las anteriores")	15
4.4.	<code>MoodleMulti</code> , <code>MoodleMultiProfe</code> y <code>MoodleMultiLastCh</code>	15
4.5.	<code>QuizMoodle</code> , <code>QuizMoodleLastCh</code> y <code>QuizVFMoodle</code>	16
4.6.	Ejemplo de uso de <code>QuizMoodleLastCh</code>	17
5.	Preguntas multi-parte ( <b>AMC</b> ) y <b>cloze</b> ( <b>Moodle</b> )	19
5.1.	<code>AMC_multipart</code> — Bloque <b>AMC</b> multi-parte	19
5.2.	Bloques <b>cloze</b> : <code>_ClozeMulti</code> , <code>_ClozeMultiProfe</code> , <code>_ClozeMultiLastCh</code> y <code>_ClozeBlock</code>	19

## Parte I

# Dando formato a las preguntas

# Capítulo 1

## Estructura del módulo `qbank._export`

---

`qbank/_export.py`

---

```
from qbank._quiz import *
```

```
<<Formato AMC>>  
<<Formato AMC Verdadero/Falso>>  
<<Formato AMC version Profe>>  
<<Formato AMC con lastchoice>>  
<<Formato AMC y multicolumna>>  
<<Formato AMC y multicolumna version Profe>>  
<<Formato AMC con lastchoice y multicolumna>>  
  
<<Método codchar para codificar caracteres no ASCII en LaTeX>>  
  
<<Quiz para Moodle>>  
<<Quiz para Moodle con valores>>  
<<Quiz para Moodle versión Profe>>  
<<Quiz para Moodle versión Profe con valores>>  
<<Quiz VF para Moodle>>  
<<Formato Moodle de preguntas de elección múltiple>>  
<<Formato Moodle de preguntas de elección múltiple para el Profe>>  
  
<<Quiz para Moodle con LastChoice>>  
<<Quiz para Moodle con LastChoice con valores>>  
<<Quiz VF para Moodle con LastChoice>>  
<<Formato Moodle de preguntas de elección múltiple con LastChoice>>  
  
<<AMC multi-parte>>  
<<Moodle cloze multi-parte>>
```

---

## Capítulo 2

# Preguntas en L<sup>A</sup>T<sub>E</sub>X para usar con Auto-Multiple-Choice

Para entender el comportamiento de las funciones de exportación, primero debemos conocer la estructura de una pregunta de opción múltiple en AMC.

Prestemos atención al siguiente ejemplo canónico extraído de la documentación oficial de [Auto Multiple Choice \(AMC\)](#):

---

```
\element{general}{
  \begin{questionmult}{pref}
    Entre las siguientes ciudades, ¿cuáles son prefecturas francesas?
    \begin{choices}
      \correctchoice{Poitiers}
      \wrongchoice{Sainte-Menehould}
      \correctchoice{Avignon}
    \end{choices}
  \end{questionmult}
}
```

---

Como se puede observar, cada pregunta se encapsula dentro del comando `\element{}{}`, el cual exige dos argumentos fundamentales:

**Primer argumento:** Una etiqueta alfanumérica que identifica el grupo temático o categoría de la pregunta (por ejemplo, `general`, o etiquetas técnicas como `TestSignificacionIndividual` o `Diagonalizacion`). Agrupar variantes bajo un mismo identificador temático permite a AMC confeccionar exámenes equilibrados seleccionando aleatoriamente un número determinado de ejercicios de cada bloque. Utilizaremos esta etiqueta como el nexo común para todas las variantes generadas por nuestras clases.

**Segundo argumento:** El bloque de la pregunta en sí. En nuestro sistema, este argumento se alimentará dinámicamente con la variante concreta obtenida al iterar `ProblemaTipo`, estructurada bajo el entorno `questionmult` de L<sup>A</sup>T<sub>E</sub>X.

Dentro del entorno `questionmult` intervienen tres componentes:

1. Una etiqueta identificadora única para la variante concreta (en nuestro caso, utilizaremos el índice secuencial provisto por el iterador).
2. El enunciado del ejercicio, compuesto por la concatenación de los textos (`self.e`) de los objetos `Supuesto` que se hayan combinado al iterar `ProblemaTipo`.
3. El entorno `choices`, que alberga las opciones a evaluar. Aquí se vuelcan los textos (`self.e`) de los objetos `Cuestion` seleccionados al iterar `ProblemaTipo`. Si la cuestión es lógicamente verdadera para la combinación de supuestos, se imprimirá como argumento del comando `\correctchoice{}`; si es falsa, se imprimirá como argumento del comando `\wrongchoice{}`.

El objetivo de este módulo es procesar las variantes generadas por los iteradores de Python y transformarlas de forma automatizada en cadenas de texto con código L<sup>A</sup>T<sub>E</sub>X válido para AMC.

## 2.1. Escritura de las cuestiones dentro del entorno choices

### 2.1.1. Texto de las cuestiones en función de si son verdaderas o falsas

Recordemos que al ejecutar:

---

```
# 1. Instanciación del iterador
variantes = iter(ProblemaTipo(ejercicio))

# 2. Extracción y desempaquetado de una variante
id_pregunta, EnunciadoCompleto, lista_Cuestiones = next(variantes)
```

---

`id_pregunta` es un identificador único o índice para la variante generada (por ejemplo, '3').

`EnunciadoCompleto` es el texto del enunciado principal de la variante (por ejemplo, 'Considere B y que B equivale a D. Conteste: ').

`lista_Cuestiones` es una lista de tuplas, donde cada tupla representa una de las opciones o subcuestiones a evaluar para esta variante. Por ejemplo [( '¿Se da D?', True, 1, '' ), ( '¿Se dan C y D?', False, 1, '' )]

Cada tupla `c` de la `lista_Cuestiones` está estructurada de la siguiente forma:

```
('Texto de la opción/cuestión', True/False, 1/0, 'explicación si ha sido incluida')
```

Para escribir las cuestiones dentro del entorno de  $\text{\LaTeX}$  `choices`, explotaremos la estructura de cada tupla `c`:

- Para asegurar la legibilidad del fichero  $\text{\LaTeX}$  generado, el código formatea la salida aplicando una indentación fija de siete espacios.
- Se evalúa la veracidad en `c[1]` para seleccionar el comando adecuado
- Finalmente se cierra la llave correspondiente al comando `}` y se inserta un salto de línea (`\n`).

---

```
Escritura del texto de la cuestión en función de su veracidad
s = s + ( ' ' * 7 ) + ( '\\correctchoice{ ' if c[1] else '\\wrongchoice { ' ) + c[0] + ' }\n'
```

---

### 2.1.2. Última opción por defecto (`lastchoices`)

AMC permite fijar opciones de respuesta al final de la lista mediante el comando `\lastchoices`, impidiendo que el motor de barajado las mezcle con las demás opciones. Esto es muy útil para introducir la clásica opción de escape cuando ninguna de las alternativas presentadas es lógicamente válida.

Examinemos este comportamiento en el siguiente fragmento de la documentación de AMC:

---

```
\begin{questionmult}{number}
  ¿Cuántos?
  \begin{choiceshoriz}
    \wrongchoice{ninguno}
    \correctchoice{uno}
    \wrongchoice{dos}
    \wrongchoice{tres}
    \lastchoices
    \correctchoice{no tanto}
    \wrongchoice{mucho}
  \end{choiceshoriz}
\end{questionmult}
```

---

En nuestras funciones, el argumento `OpcPorDefecto` (cuyo valor inicial predeterminado es 'Ninguna de las anteriores') se inyectará tras el delimitador `\lastchoices`. Su veracidad se evalúa dinámicamente: si al menos una de las cuestiones de la lista evaluada resultó ser verdadera (`True`), la opción por defecto se marcará automáticamente como `\wrongchoice`. Si todas las cuestiones de la variante fueron falsas, se convertirá en la opción `\correctchoice`.

---

```
Inclusión de lastchoice
s = s + ( ' ' * 7 ) + '\\lastchoices\n'
s = s + ( ' ' * 7 ) + ( '\\wrongchoice { ' if any([c[1] for c in cuestiones]) else '\\correctchoice{ ' ) + OpcPorDefecto + ' }\n'
```

---

## 2.2. Información sobre las cuestiones descartadas (para revisión del profesor)

AMC permite crear una copia del examen (o del banco completo de preguntas) con las respuestas correctas ya cumplimentadas en el PDF. Pensando en la posibilidad de distribuir estas copias a los alumnos, ofrece el entorno `\explain` para añadir explicaciones adicionales que les ayuden a comprender mejor el motivo por el que unas opciones son correctas y otras no. El texto contenido en este bloque queda oculto en los exámenes de los estudiantes, pero se compila y muestra de forma explícita en los documentos de tipo **Solución** o **Catálogo**.

Veamos un ejemplo de uso extraído de la documentación oficial:

---

```
\begin{question}{explanation}
  Which has the highest elevation among the following?
  \begin{choices}
    \correctchoice{Sagarmatha}
    \wrongchoice{K2}
    \wrongchoice{Mont Blanc}
    \wrongchoice{Aconcagua}
  \end{choices}
  \explain{Sagarmatha which literally means `Head of sky' is the native name of Mount Everest,
    the highest mountain in the world.}
\end{question}
```

---

Nosotros usaremos `\explain` con un propósito distinto: facilitar las tareas de auditoría y revisión de los bancos de preguntas. Cuando el tercer elemento de la tupla de la cuestión sea igual a cero (`c[2] == 0`), indicando que la pregunta ha sido descartada por el incumplimiento de una precondición, la función no la enviará al entorno de respuestas, sino que concatenará su texto y el motivo del descarte en una cadena global de diagnóstico (`ex`) que se volcará dentro del comando `\explain{}`.

---

```
ex = ex + ' cuestion: ' + c[0] + '; ' + str(c[1]) + '\n'
```

---

## Capítulo 3

# Escritura del código L<sup>A</sup>T<sub>E</sub>X para las preguntas en formato AMC

A continuación se definen las funciones principales encargadas de la exportación de código al formato L<sup>A</sup>T<sub>E</sub>X usado en AMC. Comparten una interfaz homogénea basada en los siguientes argumentos:

**nombre** Cadena en formato ASCII puro (sin espacios) que identifica la categoría macro del banco de ítems (p. ej., L-01-EcNormales-RangoX). Se utiliza como el identificador del comando `\element`.

**etiqueta** Identificador único de la variante concreta (generalmente numérico o alfanumérico compuesto).

**enunciado** El bloque de texto principal del problema generado por el iterador.

**cuestiones** Lista de tuplas con las cuestiones procesadas.

**opc** Lista opcional de control. Su primer elemento (`InstruccionesAux`) permite inyectar comandos de preámbulo locales o ajustes micro de L<sup>A</sup>T<sub>E</sub>X dentro de `\element`. El segundo elemento define el texto literal de la opción `OpcPorDefecto` (para los casos donde se use `\lastchoices`).

### 3.1. Preguntas de opción múltiple en formato para exámenes

#### 3.1.1. Función AMC

Genera la estructura estándar para exámenes impresos con un formato de respuestas a una sola columna y sin opción por defecto final (i.e., sin `\lastchoices`).

---

```
Formato AMC
def AMC (nombre, etiqueta, enunciado, cuestiones, opc=["", ""]):
    InstruccionesAux = opc[0]
    OpcPorDefecto    = opc[1]
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
    s = s + ' \\begin{questionmult}' + nombre + '- ' + str(etiqueta) + '}\n'
    s = s + ' ' + enunciado + '\n'

    s = s + ' \\begin{choices}\n'
    for c in cuestiones:
        if c[2]:
            <<Escritura del texto de la cuestión en función de su veracidad>>
    s = s + ' \\end{choices}\n'
    s = s + ' \\end{questionmult} '
    s = s + '}\n\n'
    return s
```

---

#### 3.1.2. Función AMClastCh

Similar a la función anterior, pero añade forzosamente la cláusula `\lastchoices` y evalúa la opción por defecto final para garantizar una vía de salida en la variante.

---

```
Formato AMC con lastchoice
def AMClastCh (nombre, etiqueta, enunciado, cuestiones, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto    = opc[1]
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
```

---

```

s = s + ' \\begin{questionmult}{' + nombre + '-' + str(etiqueta) + '}\n'
s = s + ' ' + enunciado + '\n'

s = s + ' \\begin{choices}\n'
for c in cuestiones:
    if c[2]:
        <<Escritura del texto de la cuestión en función de su veracidad>>
<<Inclusión de lastchoice>>
s = s + ' \\end{choices}\n'

s = s + ' \\end{questionmult} '
s = s + '}\n\n'
return s

```

---

### 3.1.3. Función AMCmc

Esta variante introduce el entorno `multicols` para distribuir las opciones de respuesta en múltiples columnas, optimizando el espacio vertical del examen impreso cuando las respuestas son cortas. Activa de forma automática la propiedad `\AMCBoxedAnswers` (véase la documentación oficial de [Auto Multiple Choice \(AMC\)](#)).

---

```

Formato AMC y multicolumna
def AMCmc (nombre, etiqueta, enunciado, cuestiones, ncols, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto = opc[1]
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
    s = s + ' \\begin{questionmult}{' + nombre + '-' + str(etiqueta) + '}\n'
    s = s + ' ' + enunciado + '\n'
    s = s + ' \\begin{multicols}' + str(ncols) + '}\AMCBoxedAnswers\n'
    s = s + ' \\begin{choices}\n'
    for c in cuestiones:
        if c[2]:
            s = s + (' ' * 7) + ('\\correctchoice{' if c[1] else '\\wrongchoice {' + c[0] + '})\n'
    s = s + ' \\end{choices}\n'
    s = s + ' \\end{multicols}\n'
    s = s + ' \\end{questionmult} '
    s = s + '}\n\n'
    return s

```

---

### 3.1.4. Función AMClastChmc

Combina las ventajas de la distribución en múltiples columnas (`c`) con la seguridad de la opción por defecto inyectada mediante `\lastchoices`.

---

```

Formato AMC con lastchoice y multicolumna
def AMClastChmc (nombre, etiqueta, enunciado, cuestiones, ncols, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto = opc[1]
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
    s = s + ' \\begin{questionmult}{' + nombre + '-' + str(etiqueta) + '}\n'
    s = s + ' ' + enunciado + '\n'
    s = s + ' \\begin{multicols}' + str(ncols) + '}\AMCBoxedAnswers\n'
    s = s + ' \\begin{choices}\n'
    for c in cuestiones:
        if c[2]:
            s = s + (' ' * 7) + ('\\correctchoice{' if c[1] else '\\wrongchoice {' + c[0] + '})\n'
    s = s + (' ' * 7) + '\\lastchoices\n'
    s = s + (' ' * 7) + ('\\wrongchoice {' if any([c[1] for c in cuestiones]) else '\\correctchoice{' + OpcPorDefecto + '})\n'
    s = s + ' \\end{choices}\n'
    s = s + ' \\end{multicols}\n'
    s = s + ' \\end{questionmult} '
    s = s + '}\n\n'
    return s

```

---

## 3.2. Preguntas de opción múltiple en formato para revisión

Estas funciones (similares a `AMC` y a `AMCmc`) tienen el propósito de generar una hoja de revisión del banco de preguntas para el docente.

- Muestran las cuestiones no válidas mediante su inclusión explícita en el entorno `\explain{}`.
- Al no requerir la opción de descarte directo para los estudiantes, prescinden del uso de `\lastchoices`.

### 3.2.1. Funciones AMCProfe y AMCmcProfe

---

```

Formato AMC version Profe
def AMCProfe (nombre, etiqueta, enunciado, cuestiones, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto = opc[1]
    ex = ''
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\\n'
    s = s + '\\begin{questionmult}' + nombre + '-' + str(etiqueta) + '}' + '\\n'
    s = s + ' ' + enunciado + '\\n'

    s = s + '\\begin{choices}\\n'
    for c in cuestiones:
        if c[2]:
            <<Escritura del texto de la cuestión en función de su veracidad>>
        else:
            <<Registro del motivo de rechazo de la cuestión>>
    s = s + '\\end{choices}\\n'
    if ex:
        s = s + '\\explain{' + ex + '}' + '\\n'
    s = s + '\\end{questionmult}' + '\\n'
    s = s + '\\n'
    return s

```

---

```

Formato AMC y multicolumna version Profe
def AMCmcProfe (nombre, etiqueta, enunciado, cuestiones, ncols, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto = opc[1]
    ex = ''
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\\n'
    s = s + '\\begin{questionmult}' + nombre + '-' + str(etiqueta) + '}' + '\\n'
    s = s + ' ' + enunciado + '\\n'
    s = s + '\\begin{multicols}' + str(ncols) + '\\\\AMCBoxedAnswers\\n'
    s = s + '\\begin{choices}\\n'
    for c in cuestiones:
        if c[2]:
            s = s + (' ' * 7) + ('\\correctchoice{' if c[1] else '\\wrongchoice {' + c[0] + '}' + '\\n'
        else:
            ex = ex + ' cuestion: ' + c[0] + '; ' + str(c[1]) + '\\n'
    s = s + '\\end{choices}\\n'
    s = s + '\\end{multicols}\\n'
    if ex:
        s = s + '\\explain{' + ex + '}' + '\\n'
    s = s + '\\end{questionmult}' + '\\n'
    s = s + '\\n'
    return s

```

---

### 3.3. Ejemplo de uso

A continuación se presenta un ejemplo completo de diseño de una pregunta combinatoria orientada a evaluar la comprensión de las **Ecuaciones Normales de MCO** y las consecuencias del rango de la matriz de regresores  $\mathbf{X}$  (presencia o ausencia de multicolinealidad perfecta).

Para mayor claridad, el problema se divide en un bloque con los elementos que permiten componer cada enunciado y otro bloque con el repertorio de cuestiones distribuidas en tres sublistas.

---

```

BloqueDeEnunciado = [
    r"En el ajuste MCO:\; $\ajusteMLG$\; ",
    [
        Supuesto(r"donde $\Matdim{X}{N}{k}$, si $\rg{\Mat{X}} < k$, con $k \leq N$,", v("Mcoli")),
        Supuesto(r"si $(\MTM{X})$ es singular,", v("Mcoli")),
        Supuesto(r"donde $\Matdim{X}{N}{k}$, si $\rg{\Mat{X}} = k$, con $k \leq N$,", -v("Mcoli")),
        Supuesto(r"si $(\MTM{X})$ es invertible,", -v("Mcoli")),
    ],
    " entonces: ",
]

BloqueDeCuestiones = [
    [
        Cuestion("ningún regresor es combinación lineal del resto", -v("Mcoli"),
            exp=r"cuando $(\MTM{X})$ es invertible, es decir, cuando $\rg{\Mat{X}} = k$"),
        Cuestion("las \emph{columnas} de $\Mat{X}$ son linealmente indep.", -v("Mcoli"),
            exp=r"cuando $(\MTM{X})$ es invertible, es decir, cuando $\rg{\Mat{X}} = k$"),
        Cuestion("las \emph{filas} de $\Mat{X}$ son linealmente indep.", False,
            exp=r"solo si $N = \rg{\Mat{X}}$"),
        Cuestion("algún regresor es combinación lineal del resto", v("Mcoli"),
            exp=r"cuando $(\MTM{X})$ es singular, es decir, cuando $\rg{\Mat{X}} < k$"),
    ],
    [
        Cuestion(r"$\Vect{y}$ es combinación lineal de los regresores", False, -v("Mcoli"),
            exp=r"cuando $\Vect{y}$ es de la forma $\MV{X}{\beta}$, es decir cuando $\res = \Vect{0}$"),
        Cuestion(r"$\MTM{X}\Vect{\beta} = \MTV{X}{y}$ tiene solución \emph{única}", -v("Mcoli"),
            exp=r"cuando $(\MTM{X})$ es invertible, es decir, cuando $\rg{\Mat{X}} = k$"),
    ]
]

```

```

    Cuestion(r"$\MTM{X}\Vect{\beta}=\MTV{X}{y}$ es compatible (tiene sol.)",
            exp=r"las ecuaciones normales siempre tienen solución"),
],
[
    Cuestion(r"el vector $\Estmc{\Vect{\beta}}$ está definido",
            exp=r"cuando $\MTM{X}\Vect{\beta}=\MTV{X}{y}$ tiene solución \emph{única}" ),
    Cuestion(r"el vector $\Estmc{\Vect{\beta}}$ es indeterminado",
            exp=r"cuando $\MTM{X}\Vect{\beta}=\MTV{X}{y}$ tiene infinitas soluciones"),
    Cuestion("la correlación entre algunos regresores es 1 en valor absoluto",
            exp=r"cuando algún regresor es un múltiplo de otro"),
    Cuestion("la correlación entre regresores es \emph{menor} que 1 en valor absoluto",
            exp=r"cuando los regresores son linealmente independientes"),
],]

p = [ BloqueDeEnunciado + BloqueDeCuestiones ]

directorio = "../ejemplos/"
nombre="L-01-EcNormales-RangoX"
mc=2;

preguntas = {};
preguntas[nombre] = ProblemaTipo( p )
with open(directorio + nombre + ".tex","w") as f:
    for i,nom in enumerate(preguntas):
        for var in preguntas[nom]:
            f.write( AMClastChmc(nombre, str(i)+"-"+var[0], var[1], var[2], mc, ["","Ninguna de las anteriores"]) )

```

Analicemos el ejemplo:

### 1. Modularidad en la programación del ejercicio.

- **BloqueDeEnunciado:** Se encarga en exclusiva de fijar el escenario matemático. Contiene las cadenas de texto introductorias y la lista de objetos `Supuesto`. En cada iteración, el motor seleccionará un único escenario de este bloque (por ejemplo, si la matriz es singular o invertible) para construir la hipótesis de partida de cada examen individual.
- **BloqueDeCuestiones:** Alberga las sublistas de objetos `Cuestion`. Cada sublista representa una dimensión conceptual del problema (relación de los regresores, propiedades del sistema de ecuaciones y características del estimador). En cada iteración, el motor seleccionará una única cuestión de cada bloque para combinar las tres cuestiones con las hipótesis del enunciado correspondientes a la variante que ha generado.
  - Nótese que la primera cuestión de la segunda lista tiene un parámetro de activación basado en la precondition `=-v("Mcoli")=`. Esto ocasiona que esta opción (que es matemáticamente falsa para cualquier escenario) se oculte por completo y no se presente al alumno si el enunciado de la variante implica la presencia de multicolinealidad.
- **Concatenación:** Mediante la instrucción `p = [BloqueDeEnunciado + BloqueDeCuestiones]` se unifican las listas nativas de Python antes de instanciar el objeto `ProblemaTipo`, el cual pone en marcha el motor combinatorio.

### 2. El motor de evaluación mediante variables lógicas (`v`). Para analizar la veracidad de las cuestiones dadas las hipótesis elegidas por el motor, se utiliza la función de evaluación dinámica `v('Mcoli')`:

- El estado de la condición de multicolinealidad actúa como un interruptor booleano interno. Si el supuesto extraído para el examen es de multicolinealidad, `v('Mcoli')` evalúa a `True`. Si el supuesto es de rango completo por columnas, evalúa a `False`.
- El uso de operadores de negación (`-`) permite invertir este comportamiento según la semántica de la respuesta. Por ejemplo, si se elige el supuesto de invertibilidad (donde `v('Mcoli') = False=`), un ítem condicionado con `-v('Mcoli')` pasará automáticamente a considerarse como una opción válida (`-False = True=`).
- El sistema también admite valores de verdad absolutos independientes del supuesto (como `True` o `False`) para aquellas propiedades matemáticas inmutables, como el hecho de que las ecuaciones normales siempre sean compatibles.

### 3. Traducción a código $\LaTeX$ para exámenes AMC o para gestión del banco de preguntas. La parte final del script se encarga de estructurar el banco de ejercicios y generar el código fuente en $\LaTeX$ :

- **Diccionario de variantes y visualización:** El proceso comienza inicializando un diccionario vacío (`preguntas = {}`) para almacenar las variantes del ejercicio y definiendo el número de columnas para las cuestiones del examen mediante la variable `mc` (en este caso, `mc=2` para doble columna).

- **Modo Profesor frente a Modo Alumno:** Al instanciar la estructura `p`, el motor permite dos enfoques según el objetivo del documento:
  - **Para revisión del profesor:** Se utiliza `ProblemaTipoProfe(p)`, diseñado para mostrar el desglose de las soluciones y facilitar la validación manual del ejercicio (este ejemplo no lo hace).
  - **Para el examen del alumno:** Se emplea `ProblemaTipo(p)`, que compila y mezcla el banco de ítems procesando de forma sistemática todas las combinaciones que cumplan los requisitos lógicos.
  - **Nota sobre variantes numéricas:** Si el script generase además variaciones numéricas aleatorias mediante un bucle, la clave del diccionario debe indexarse de forma dinámica para no sobrescribir las variantes: `preguntas[nombre+str(numopcion)] = ProblemaTipo(p)`.
- **Volcado final con AMC:** El bucle de escritura final invoca a la función `AMClastChmc` (importada del módulo de formatos). Esta función abstrae por completo la sintaxis de  $\text{\LaTeX}$ , distribuye visualmente las opciones según el parámetro `mc` e integra automáticamente la respuesta comodín "Ninguna de las anteriores" al cierre de cada bloque de opciones.

### 3.4. Variante para preguntas Verdadero/Falso en formato para exámenes en el aula

Las preguntas aleatorizadas generadas por la clase `ProblemaVF` poseen una estructura simplificada: carecen de precondiciones de filtrado dinámico. En consecuencia, el parámetro `flag` de descarte de la tupla (`c[2]`) no existe. Las tuplas inyectadas desde dicho banco constan únicamente de dos componentes: el texto y su booleano de veracidad: '(Texto, True/False)'

A continuación se define la función específica encargada de iterar directamente esta colección binaria para renderizar la salida de examen:

---

```
Formato AMC Verdadero/Falso
```

---

```
def AMC_VF (nombre, etiqueta, enunciado, cuestiones, opc=["", "Ninguna de las anteriores"]):
    InstruccionesAux = opc[0]
    OpcPorDefecto = opc[1]
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
    s = s + ' \\begin{questionmult}' + nombre + '-' + str(etiqueta) + '}\n'
    s = s + ' ' + enunciado + '\n'

    s = s + ' \\begin{choices}\n'
    for c in cuestiones:
        s = s + (' ' * 7) + ('\\correctchoice' if c[1] else '\\wrongchoice ') + c[0] + '\n'
    s = s + ' \\end{choices}\n'

    s = s + ' \\end{questionmult} '
    s = s + '}\n\n'
    return s
```

---

#### 3.4.1. Ejemplo de uso

El siguiente bloque implementa el volcado físico del código generado por `AMC_VF` a un archivo con extensión `.tex`, creando un examen de 4 variantes compuestas por 3 preguntas de verdadero/falso seleccionadas aleatoriamente de nuestro repositorio básico.

---

```
EjemploVFamc.py
```

---

```
from qbank import *

enunciado = "Indique qué afirmaciones son verdaderas:"
bancoVF = [
    ("Todo alumno que va a clase aprueba", False),
    ("Todo alumno que suspende va a clase", False),
    ("Todo alumno que sabe aprueba", True),
    ("Si aprueban todos eres buen profesor", False),
    ("Si suspenden todos eres mal profesor", False),
    ("Si suspenden todos es frustrante", True),]
GenVar = iter( ProblemaVF (enunciado, bancoVF, 3) ) # Tres preguntas por variante

nombre = "EjemploVF"
directorio = "../ejemplos/"
with open(directorio + nombre + ".tex", "w") as f:
    for i in range(4):
        var = (next(GenVar))
        f.write(AMC_VF(nombre, var[0], var[1], var[2]))
```

---

## Capítulo 4

# Preguntas en XML para usar con Moodle

Para generar los ficheros `xml` necesarios para Moodle, seguiremos un método indirecto. Escribir directamente en XML es tedioso debido a la gestión de caracteres no ASCII y las expresiones matemáticas de  $\text{\LaTeX}$ .

La solución más práctica consiste en generar desde Python un fichero `.tex` de  $\text{\LaTeX}$  que emplee el paquete `moodle`. Al compilar dicho fichero (con `pdflatex`, `lualatex` o `xelatex`) obtendremos simultáneamente una versión en PDF del banco de preguntas (ideal para revisión) y el archivo `.xml` listo para ser importado en Moodle. Además, este enfoque nos evita lidiar con la compleja estructura del XML de Moodle, sustituyéndola por la familiar sintaxis de  $\text{\LaTeX}$ .

### 4.1. Preámbulo de los ficheros $\text{\LaTeX}$ para exportación a Moodle

Para que se genere el archivo XML, es estrictamente necesario que el preámbulo del fichero `.tex` cargue el paquete `moodle`.

El preámbulo definido a continuación está configurado para que el documento PDF resultante tenga un tamaño de fuente de `11pt` y márgenes estrechos mediante el paquete `fullpage` para optimizar la lectura del docente. Asimismo, se cargan `graphicx` y `fancyvrb` para soportar gráficos y código literal (todo esto es opcional).

Este preámbulo se almacena en una cadena de caracteres de Python (`s`) que abre el entorno `document`:

---

```
Preámbulo fichero LaTeX con paquete moodle
s = "\\documentclass[11pt]{article}\n\n"
s = s + "\\usepackage[cm,headings]{fullpage}\n\n"
s = s + "\\usepackage{moodle}\n\n"
s = s + "\\usepackage{graphicx}\n\n"
s = s + "\\usepackage{fancyvrb}\n\n"
s = s + "\\ifPDFTeX % FOR LATEX and PDFLATEX\n"
s = s + "    \\usepackage[utf8]{inputenc} % necessary\n"
s = s + "    \\usepackage[OT1]{fontenc} % necessary\n"
s = s + "\\else % assuming XELATEX or LUALATEX\n"
s = s + "    \\usepackage{fontspec}\n"
s = s + "\\fi\n\n"
s = s + auxLaTeX + "\n" + "\\newcommand\\peque{}" + "\n\n"
s = s + "\\begin{document}\n\n"
```

---

#### 4.1.1. Codificación de caracteres no ASCII

Para evitar la tediosa escritura manual de acentos y eñes al estilo  $\text{\LaTeX}$  (como `\'{a}` o `\~{n}`), delegamos la tarea en Python mediante la función `codchar(s)`, que realiza una sustitución masiva de strings:

---

```
Método codchar para codificar caracteres no ASCII en LaTeX
def codchar(s):
    s = s.replace("á", "\\'{a}")
    s = s.replace("é", "\\'{e}")
    s = s.replace("í", "\\'{i}")
    s = s.replace("ó", "\\'{o}")
    s = s.replace("ú", "\\'{u}")
    s = s.replace("ñ", "\\~{n}")
    return s
```

---

## 4.2. Estructura de preguntas de opción múltiple con el paquete moodle

La sintaxis nativa de `moodle.sty` para una pregunta de opción múltiple con una única respuesta correcta sigue la siguiente estructura:

---

```
----- Ejemplo de entorno multi para una cuestión usando el paquete moodle -----
\begin{multi}[opciones de pregunta]{nombre de pregunta}
texto de pregunta
\item[opciones]* cuestión correcta 1
\item[opciones] cuestión incorrecta 1
:
\item[opciones] cuestión incorrecta n
\end{multi}
```

---

Si la cuestión es correcta se designa mediante un asterisco (\*) inmediatamente después del comando `\item`. El orden entre las respuestas es irrelevante.

### 4.2.1. Parámetros opcionales del comando `\item`

**fraction** Controla el peso de la respuesta. Por defecto, `\item*` asigna `fraction=100` y `\item` asigna `fraction=0`. Puede modificarse manualmente para otorgar puntuaciones parciales (por ejemplo: `\item[fraction=50]`).

**feedback** Define la retroalimentación que leerá el alumno tras contestar. Debe ir entre llaves. (por ejemplo: `\item[feedback={¡Muy bien!}]`).

---

```
----- Ejemplo de uso de feedback -----
\begin{multi}[points=3]{Superhéroes}
¿Quién es el superhéroe al que más le gusta el queso?
\item[feedback={Has respondido increíblemente mal}] Mister Increíble
\item[feedback={Muy bien. ¡Tómame un quesito de mi parte!}]* Super Ratón
\item[feedback={Tu respuesta es supermala}] Superman
\end{multi}
```

---

### 4.2.2. Parámetros opcionales del entorno `multi`

**multiple** Activado mediante `multiple` o `single=false`, permite ejercicios con varias cuestiones correctas. Admite dos modos de puntuación:

1. **Modo avanzado:** El profesor define las fracciones a mano (pueden ser negativas para penalizar).
2. **Modo automático:** Se marcan las correctas con `\item*` y el paquete distribuye el 100% equitativamente entre ellas, aplicando penalizaciones automáticas a las incorrectas.

Por ejemplo, los siguientes dos ejemplos son equivalentes:

---

```
----- Ejemplo de uso de fraction -----
\begin{multi}[multiple]{Modo Automático}
¿Cuáles números son primos?
\item 2
\item* 5
\item* 7
\item 1
\item 6
\end{multi}
\begin{multi}[multiple]{Modo Avanzado}
¿Cuáles números son primos?
\item[fraction=-50] 2
\item[fraction=50] 5
\item[fraction=50] 7
\item[fraction=-50] 1
\item[fraction=-50] 6
\end{multi}
```

---

**points** Establece el valor de cada cuestión del cuestionario (por defecto 1).

**Nota sobre la puntuación en Moodle:** A diferencia de AMC, donde se puede parametrizar una esperanza nula estricta con notas globales negativas, Moodle trunca la calificación mínima de una pregunta de opción múltiple a 0. No permite notas globales negativas en estos entornos.

**TODO:** Investigar si esta limitación se puede solventar utilizando preguntas de tipo *Cloze*.

## 4.3. Escritura automatizada de las cuestiones desde Python

El formato de los textos de aclaración y las respuestas correctas varía según si generamos el banco para los alumnos o la plantilla de revisión para el docente.

Para la versión del alumno, definimos `itemBuena` e `itemMala` encargadas de inyectar el parámetro `feedback`:

---

```
Definición itemBuena e itemMala para cuestiones en Moodle
def itemBuena(aclaracion):
    return ("\\item[feedback={"+ codchar(aclaracion) + "}]* ") if aclaracion else r"\\item* "
def itemMala(aclaracion):
    return ("\\item[feedback={"+ codchar(aclaracion) + "}] ") if aclaracion else r"\\item "
```

---

Para la versión del profesor, estas funciones permiten visualizar el reparto de los puntos en función del número de cuestiones verdaderas y falsas (`numBuenas` o `numMalas`) para reflejar cómo penalizaría el formato en un diseño de examen estándar AMC:

---

```
Definición itemBuena e itemMala para cuestiones en Moodle para Profe
def itemBuena(numBuenas, aclaracion):
    return ("\\item[fraction="+ str(round(100/numBuenas)) + feedback(aclaracion) + "]") if numBuenas else "\\item*"
def itemMala(numMalas, aclaracion):
    return ("\\item[fraction="+ str(-round(100/numMalas)) + feedback(aclaracion) + "]") if numMalas else "\\item"
```

---

La siguiente función auxiliar da formato al bloque de texto de la retroalimentación:

---

```
Escritura aclaración en Moodle
def feedback(texto):
    return r",\feedback={"+codchar(texto)+"}"
```

---

### 4.3.1. Procesamiento de las variantes extraídas del iterador

Recordemos el desempaqueado de una variante generada por nuestro motor:

---

```
# 1. Instanciación del iterador
variantes = iter(ProblemaTipo(ejercicio))

# 2. Extracción y desempaqueado de una variante
id_pregunta, EnunciadoCompleto, lista_Cuestiones = next(variantes)
```

---

donde `lista_Cuestiones` es una lista de tuplas, donde cada tupla representa una de las opciones o subcuestiones a evaluar para esta variante. Por ejemplo `[('¿Se da D?', True, 1, 'Explicación para D'), ('¿Se dan C y D?', False, 1, '')]`

Cada tupla `c` de la `lista_Cuestiones` está estructurada de la siguiente forma:

```
('Texto de la opción/cuestión', True/False, 1/0, 'explicación si ha sido incluida')
```

y cada una es procesada en bucle según el destinatario:

**Para la versión del estudiante** Evaluamos la veracidad en `c[1]` para decidir el método de formato aplicando una indentación limpia de 7 espacios.

---

```
Escritura de las cuestiones para Moodle
for c in cuestiones:
    fb = c[3] if len(c) > 3 else ''
    s = s + (' ' * 7) + (itemBuena(fb) if c[1] else itemMala(fb)) + codchar(c[0]) + '\n'
```

---

**Para la versión del profesor** Controlamos además si la cuestión fue descartada por precondition (`c[2] == 0`), enviando el descarte al registro de diagnóstico global `ex`.

---

```
Escritura de las cuestiones para Moodle para el Profe
for c in cuestiones:
    if c[2]:
        s = s + (' ' * 7) + (itemBuena(b, c[3]) if c[1] else itemMala(m, c[3])) + codchar(c[0]) + '\n'
    else:
        ex = ex + ' cuestion: ' + c[0] + '; ' + str(c[1]) + '\n'
```

---

### 4.3.2. Gestión de la opción por defecto ("Ninguna de las anteriores")

Moodle exige que exista al menos una opción correcta en el entorno o la compilación fallará. Para blindar el comportamiento si la combinatoria descarta todas las respuestas correctas de una variante, añadimos dinámicamente un ítem comodín cuya veracidad es complementaria al recuento de respuestas verdaderas.<sup>1</sup> El modo de hacerlo es añadir una tupla más en la lista `cuestiones`, que será falsa si en el resto de tuplas una o más cuestiones son verdaderas.

---

```
_____ cuestion alternativa por defecto para Moodle _____  
v = [c[1] for c in cuestiones].count(True)  
cuestiones = cuestiones + [(codchar(lastchoice), (False if v else True), 1, '')]
```

---

### 4.4. MoodleMulti, MoodleMultiProfe y MoodleMultiLastCh

**MoodleMulti** Genera una cadena de caracteres con los comandos L<sup>A</sup>T<sub>E</sub>X correspondientes a una pregunta de opción múltiple dentro del entorno `{multi}`. Es la función equivalente a la función AMC (sección 3.1.1) para las preguntas en formato AMC.

---

```
_____ Formato Moodle de preguntas de elección múltiple _____  
def MoodleMulti (nombre, variante, enunciado, cuestiones):  
<<Escritura del entorno multi>>
```

---

El método escribe el entorno aplicando los parámetros opcionales `multiple` y `points` (con valor igual al número de opciones de respuesta de la pregunta; es decir, la longitud de la lista `cuestiones`):

---

```
_____ Escritura del entorno multi _____  
<<Definición itemBuena e itemMala para cuestiones en Moodle>>  
  
ex = ''  
s = " \\begin{multi}[multiple, points=" + str(len(cuestiones)-1) + "]"  
s = s + "{ " + codchar(nombre) + "-" + str(variante) + "}\n"  
s = s + " " + enunciado + "\n"  
  
<<Escritura de las cuestiones para Moodle>>  
  
s = s + " \\end{multi}\n\n"  
return s
```

---

**MoodleMultiLastCh** Es similar a la función `MoodleMulti`, pero añadiendo la opción por defecto antes de generar el entorno.

---

```
_____ Formato Moodle de preguntas de elección múltiple con LastChoice _____  
def MoodleMultiLastCh (nombre, variante, enunciado, cuestiones, \  
    lastchoice="Las demás opciones son falsas"):  
<<cuestion alternativa por defecto para Moodle>>  
<<Escritura del entorno multi>>
```

---

**MoodleMultiProfe** Versión homóloga para la plantilla del docente. Realiza previamente el conteo de respuestas correctas (`b`) e incorrectas (`m`) para mostrar el porcentaje de puntuación de cada cuestión en el documento de revisión.

---

```
_____ Formato Moodle de preguntas de elección múltiple para el Profe _____  
def MoodleMultiProfe (nombre, variante, enunciado, cuestiones):  
<<Escritura aclaración en Moodle>>  
<<Escritura del entorno multi para el Profe>>
```

---

Para calcular el valor para el argumento `fraction` del comando `\item`, necesitamos contar, entre las cuestiones no descartadas (`c[2]=1`), cuantas de cuestiones verdaderas (variable `b`) y cuántas son falsas (variable `m`).

---

```
_____ Conteo de respuestas correctas e incorrectas _____  
b = 0; m = 0;  
for c in cuestiones:  
    if c[2]:  
        if c[1]:  
            b+=1  
        else:  
            m+=1
```

---

<sup>1</sup>Moodle no dispone de un equivalente a `\lastchoice` para fijar su posición al barajar, por lo que el texto recomendado es *"Las demás opciones son falsas"*.

La estructura del código que escribe el entorno `{multi}` es similar al de las dos funciones anteriores:

---

```
_____ Escritura del entorno multi para el Profe _____
<<Conteo de respuestas correctas e incorrectas>>
<<Definición itemBuena e itemMala para cuestiones en Moodle para Profe>>

ex = ''
s = " \\begin{multi}[multiple, fractiontol=5.1" + "]"
s = s + "{ " + codchar(nombre) + "-" + str(variante) + " }\n"
s = s + " " + enunciado + "\n"

<<Escritura de las cuestiones para Moodle para el Profe>>

s = s + " \\end{multi}\n\n"
return s
```

---

## 4.5. QuizMoodle, QuizMoodleLastCh y QuizVFMoodle

Estas funciones de alto nivel abren el flujo del archivo con `with open()`, vuelcan la cadena del preámbulo, iteran sobre el diccionario de problemas inyectando las variantes y cierran el entorno principal `quiz`. Soporta la integración de motores de renderizado de plantillas (como [Jinja2](#)) si se requiere sustitución de variables.

Para asegurar que la entrada sea procesada correctamente, se utiliza un método auxiliar que encapsula objetos sueltos en diccionarios:

---

```
_____ Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario _____
def creaDiccionario(x, key='key'):
    return x if isinstance(x, dict) else {key: x}
```

---

A continuación se despliegan las implementaciones de los distintos exportadores del módulo:

---

```
_____ Quiz para Moodle _____
def QuizMoodle (nombre, directorio, problema, opc=["", ""]):
    <<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
    problema = creaDiccionario(problema, nombre)
    auxLaTeX = opc[0]
    <<Preámbulo fichero LaTeX con paquete moodle>>
    s = s + "\\begin{quiz}{ " + nombre + " }\n\n"
    with open(directorio + nombre + ".tex", "w") as f:
        f.write(s)
        for i, nom in enumerate(problema):
            for etiqueta, partes in problema[nom].por_partes():
                f.write(_ClozeBlock(nom, etiqueta, partes, _ClozeMulti))
        f.write("\\end{quiz}\n\n\\end{document}\n")
```

---

---

```
_____ Quiz para Moodle con valores _____
def QuizMoodleConVariables (nombre, directorio, problema, environment, Valores, opc=["", ""]):
    <<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
    problema = creaDiccionario(problema, nombre)
    auxLaTeX = opc[0]
    <<Preámbulo fichero LaTeX con paquete moodle>>
    s = s + "\\begin{quiz}{ " + nombre + " }\n\n"
    with open(directorio + nombre + ".tex", "w") as f:
        f.write(s)
        for i, nom in enumerate(problema):
            for etiqueta, partes in problema[nom].por_partes():
                template = environment.from_string(_ClozeBlock(nom, etiqueta, partes, _ClozeMulti))
                content = template.render(Valores())
                f.write(content)
        f.write("\\end{quiz}\n\n\\end{document}\n")
```

---

---

```
_____ Quiz para Moodle versión Profe _____
def QuizMoodleProfe (nombre, directorio, problema, opc=["", ""]):
    <<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
    problema = creaDiccionario(problema, nombre)
    auxLaTeX = opc[0]
    <<Preámbulo fichero LaTeX con paquete moodle>>
    s = s + "\\begin{quiz}{ " + nombre + " }\n\n"
    with open(directorio + nombre + ".tex", "w") as f:
        f.write(s)
        for i, nom in enumerate(problema):
            for etiqueta, partes in problema[nom].por_partes():
                f.write(_ClozeBlock(nom, etiqueta, partes, _ClozeMultiProfe))
        f.write("\\end{quiz}\n\n\\end{document}\n")
```

---

---

```
_____ Quiz para Moodle versión Profe con valores _____
def QuizMoodleProfeConVariables (nombre, directorio, problema, environment, Valores, opc=["", ""]):
    <<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
```

---

```

problema = creaDiccionario(problema, nombre)
auxLaTeX = opc[0]
<<Preámbulo fichero LaTeX con paquete moodle>>
s = s + "\\begin{quiz}" + nombre + "}\n\n"
with open(directorio + nombre + ".tex", "w") as f:
    f.write(s)
    for i,nom in enumerate(problema):
        for etiqueta, partes in problema[nom].por_partes():
            template = environment.from_string(ClozeBlock(nom, etiqueta, partes, _ClozeMultiProfe))
            content = template.render(Valores())
            f.write(content)
    f.write("\\end{quiz}\n\n\\end{document}\n")

```

---

Quiz para Moodle con LastChoice

```

def QuizMoodleLastCh (nombre, directorio, problema, opc=["", "Las demás opciones son falsas"]):
<<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
problema = creaDiccionario(problema, nombre)
auxLaTeX = opc[0]
OpcPorDefecto = opc[1]
<<Preámbulo fichero LaTeX con paquete moodle>>
s = s + "\\begin{quiz}" + nombre + "}\n\n"
with open(directorio + nombre + ".tex", "w") as f:
    f.write(s)
    for i,nom in enumerate(problema):
        for etiqueta, partes in problema[nom].por_partes():
            f.write(_ClozeBlock(nom, etiqueta, partes, lambda c: _ClozeMultiLastCh(c, opc[1])))
    f.write("\\end{quiz}\n\n\\end{document}\n")

```

---

Quiz para Moodle con LastChoice con valores

```

def QuizMoodleLastChConVariables (nombre, directorio, problema, environment, Valores, opc=["", "Las demás opciones son falsas"]):
<<Método auxiliar que transforma un objeto en diccionario si no era ya un diccionario>>
problema = creaDiccionario(problema, nombre)
auxLaTeX = opc[0]
OpcPorDefecto = opc[1]
<<Preámbulo fichero LaTeX con paquete moodle>>
s = s + "\\begin{quiz}" + nombre + "}\n\n"
with open(directorio + nombre + ".tex", "w") as f:
    f.write(s)
    for i,nom in enumerate(problema):
        for etiqueta, partes in problema[nom].por_partes():
            _ClozeBlock(nom, etiqueta, partes, lambda c: _ClozeMultiLastCh(c, opc[1]))
            content = template.render(Valores())
            f.write(content)
    f.write("\\end{quiz}\n\n\\end{document}\n")

```

**Bancos Verdadero/Falso (ProblemaVF)** Las funciones homólogas extraen variantes basadas en un iterador clásico mediante el método 'next()', permitiendo fijar el número exacto de preguntas deseadas:

---

Quiz VF para Moodle

```

def QuizVFMoodle (nombre, directorio, GenVar, num, opc=["", ""]):
auxLaTeX = opc[0]
<<Preámbulo fichero LaTeX con paquete moodle>>
s = s + "\\begin{quiz}" + nombre + "}\n\n"
with open(directorio + nombre + ".tex", "w") as f:
    f.write(s)
    for i in range(num):
        var = next(GenVar)
        f.write( MoodleMulti (codchar(nombre), var[0], codchar(var[1]), var[2]) )
    f.write("\\end{quiz}\n\n\\end{document}\n")

```

## 4.6. Ejemplo de uso de QuizMoodleLastCh

A continuación se ilustra la ejecución de scripts independientes para generar tanto un bloque relacional condicionado (ProblemaTipo) como una batería aleatoria estructurada desde una lista estática de enunciados (ProblemaVF):

---

EjemploMoodle.py

```

from qbank import *
p = ProblemaTipo([
    "Considere ",
    [
        Supuesto("$\\mathcal{A}$ ", v("A")),
        Supuesto("$\\mathcal{B}$ ", v("B")),
    ],
    [
        Supuesto("y que $\\mathcal{A}\\rightarrow\\mathcal{C}$.", v("A") >> v("C")),
        Supuesto("y que $\\mathcal{B}\\Leftarrow\\mathcal{D}$.", v("B") ** v("D")),
    ],
])

```

```

    ],
    "Indique qué opción es correcta: ",
    [
        Cuestion("Entonces  $\mathcal{C}$  es verdadero", v("C")),
        Cuestion("Entonces  $\mathcal{D}$  es verdadero", v("D")),
        Cuestion("Entonces  $\mathcal{E}$  es verdadero", v("E"), v("D")),
    ],
])
nombre = "EjemploMoodle"
directorio = "../ejemplos/"
QuizMoodleLastCh(nombre, directorio, p)

```

---

EjemploVFMoodle.py

---

```

from qbank import *

```

```

enunciado = "Indique qué afirmaciones son verdaderas:"
banco = [
    ("Todo alumno que va a clase aprueba", False),
    ("Todo alumno que suspende va a clase", False),
    ("Todo alumno que sabe aprueba", True),
    ("Si aprueban todos eres buen profesor", False),
    ("Si suspenden todos eres mal profesor", False),
    ("Si suspenden todos es frustrante", True),
]

b = [(codchar(p[0]), p[1]) for p in banco]
GenVar = iter(ProblemaVF(codchar(enunciado), b, 3))

nombre = "EjemploVFMoodle"
directorio = "../ejemplos/"
QuizVFMoodle(nombre, directorio, GenVar, 4)

```

---

## Capítulo 5

# Preguntas multi-parte (AMC) y cloze (Moodle)

Los formatos AMC y Moodle permiten agrupar varias sub-preguntas de opción múltiple bajo un único enunciado general. Cada sub-pregunta tiene su propio texto introductorio y su propio bloque de opciones.

### 5.1. AMC\_multipart — Bloque AMC multi-parte

Genera un único bloque `\begin{questionmult}` que contiene múltiples grupos de opciones. Cada grupo está precedido por `\emph{intro}` y envuelto en `\AMCnoCompleteMulti` (suprime la opción comodín en ese sub-bloque).

---

```
AMC multi-parte
def AMC_multipart(nombre, etiqueta, enunciado, subpreguntas, opc=[""]):
    """Genera el bloque AMC para una pregunta con enunciado común y sub-preguntas.

    subpreguntas: list de (intro, [(texto, correcto, activa, exp), ...])
    Cada sub-pregunta produce un bloque \begin{choices} independiente precedido
    por \emph{intro} y envuelto en \AMCnoCompleteMulti.
    """
    InstruccionesAux = opc[0] if opc else ""
    s = '\\element{' + nombre + '}' + InstruccionesAux + '\n'
    s += ' \\begin{questionmult}' + nombre + '-' + str(etiqueta) + '}\n'
    s += ' ' + enunciado + '\n\n'
    for intro, cuestiones in subpreguntas:
        s += ' \\emph{' + intro + '}\n'
        s += ' {\\AMCnoCompleteMulti\n'
        s += ' \\begin{choices}\n'
        for c in cuestiones:
            s += ' ' * 5 + ('\\correctchoice{' if c[1] else '\\wrongchoice {') + c[0] + '}\n'
        s += ' \\end{choices}\n'
        s += ' }\n\n'
    s += ' \\end{questionmult} '
    s += '}\n\n'
    return s
```

---

Se usa en un bucle igual que AMC:

---

```
with open("preguntas.tex", "w") as f:
    for etiqueta, enunciado, subpreguntas in p:
        f.write(AMC_multipart("MiCuestionario", etiqueta, enunciado, subpreguntas))
```

---

### 5.2. Bloques cloze: `_ClozeMulti`, `_ClozeMultiProfe`, `_ClozeMultiLastCh` y `_ClozeBlock`

Toda la exportación a Moodle usa el entorno *cloze*, tenga el ejercicio una o varias partes. Cada variante produce una pregunta `\begin{cloze}{nombre-etiqueta}` que contiene un entorno `\begin{multi}` por cada parte (sin nombre, como exige el paquete `moodle` para los entornos embebidos). El feedback por opción, las fracciones del modo profe y el last-choice funcionan dentro de *cloze* (manual de `moodle.sty`

§2.4.7 y documentación de Moodle), por lo que reutilizamos exactamente la misma lógica de escritura de `\item` que las preguntas sueltas.

`_ClozeMulti`, `_ClozeMultiProfe` y `_ClozeMultiLastCh` generan el bloque `\begin{multi}...\end{multi}` de una parte (versión alumno, profe y last-choice respectivamente); `_ClozeBlock` envuelve las partes de una variante en el entorno `\begin{cloze}`.

---

```

Moodle cloze multi-parte
def _ClozeMulti(cuestiones):
    <<Definición itemBuena e itemMala para cuestiones en Moodle>>
    s = " \\begin{multi}[multiple, points=" + str(len(cuestiones)-1) + "]\n"
    <<Escritura de las cuestiones para Moodle>>
    s = s + " \\end{multi}\n\n"
    return s

def _ClozeMultiLastCh(cuestiones, lastchoice="Las demás opciones son falsas"):
    <<question alternativa por defecto para Moodle>>
    <<Definición itemBuena e itemMala para cuestiones en Moodle>>
    s = " \\begin{multi}[multiple, points=" + str(len(cuestiones)-1) + "]\n"
    <<Escritura de las cuestiones para Moodle>>
    s = s + " \\end{multi}\n\n"
    return s

def _ClozeMultiProfe(cuestiones):
    <<Escritura aclaración en Moodle>>
    <<Conteo de respuestas correctas e incorrectas>>
    <<Definición itemBuena e itemMala para cuestiones en Moodle para Profe>>
    ex = ''
    s = " \\begin{multi}[multiple, fractiontol=5.1]\n"
    <<Escritura de las cuestiones para Moodle para el Profe>>
    s = s + " \\end{multi}\n\n"
    return s

def _ClozeBlock(nombre, etiqueta, partes, cuerpo):
    """Envuelve las partes de una variante en un entorno cloze.
    `cuerpo(cuestiones)` genera el bloque \\begin{multi}...\end{multi} de cada parte."""
    s = " \\begin{cloze}{" + codchar(nombre) + "-" + str(etiqueta) + "}\n"
    for enunciado, cuestiones in partes:
        s += " " + codchar(enunciado) + "\n"
        s += cuerpo(cuestiones)
    s += " \\end{cloze}\n\n"
    return s

```

---