

# 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.	Helpers: identificadores y codificación de caracteres	5
2.1.1.	Función <code>_sanitize_name</code>	5
2.1.2.	Función <code>codchar</code>	5
2.2.	Escritura de las cuestiones dentro del entorno <code>choices</code>	5
2.2.1.	Texto de las cuestiones en función de si son verdaderas o falsas	5
2.2.2.	Última opción por defecto ( <code>lastchoices</code> )	6
2.3.	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.	Función unificada <code>AMCblock</code>	7
3.2.	Variante para preguntas Verdadero/Falso en formato para exámenes en el aula	8
3.2.1.	Ejemplo de uso	9
3.3.	Ejemplo de uso de <code>AMCblock</code>	9
4.	Preguntas en <b>XML</b> para usar con <b>Moodle</b>	10
4.1.	Preámbulo de los ficheros $\text{\LaTeX}$ para exportación a Moodle	10
4.2.	Estructura de preguntas de opción múltiple con el paquete <code>moodle</code>	11
4.2.1.	Parámetros opcionales del comando <code>\item</code>	11
4.2.2.	Parámetros opcionales del entorno <code>multi</code>	11
4.3.	Escritura automatizada de las cuestiones desde Python	12
4.3.1.	Procesamiento de las variantes extraídas del iterador	12
4.3.2.	Gestión de la opción por defecto ("Ninguna de las anteriores")	13
4.4.	<code>MoodleMulti</code> , <code>MoodleMultiProfe</code> y <code>MoodleMultiLastCh</code>	13
4.5.	<code>QuizMoodle</code> y <code>QuizVFMoodle</code>	14
4.5.1.	<code>QuizMoodle</code> — función unificada	14
4.5.2.	<code>QuizMoodle</code> — función principal	14
4.5.3.	<code>QuizMoodleProfe</code> — versión docente	15
4.5.4.	<code>QuizVFMoodle</code>	15
4.6.	Ejemplo de uso de <code>QuizMoodle</code>	15
5.	Preguntas multi-parte ( <b>AMC</b> ) y cloze ( <b>Moodle</b> )	17
5.1.	<code>AMC_multipart</code> — Bloque <b>AMC</b> multi-parte	17
5.2.	Bloques cloze: <code>_ClozeMulti</code> , <code>_ClozeMultiProfe</code> , <code>_ClozeMultiLastCh</code> y <code>_ClozeBlock</code>	17

## Parte I

# Dando formato a las preguntas

# Capítulo 1

## Estructura del módulo `qbank._export`

---

```
qbank/_export.py
import re as _re
from qbank._quiz import *

# Helpers

<<Helper _sanitize_name>>
<<Método codchar para codificar caracteres no ASCII en LaTeX>>

# Exportadores AMC

<<AMCblock>>
<<Formato AMC Verdadero/Falso>>
<<AMC multi-parte>>

# Exportadores Moodle

<<_MOODLE_HEADER y _moodle_header>>
<<Quiz para Moodle>>
<<Quiz para Moodle versión Profe>>
<<Quiz VF para Moodle>>
<<Formato Moodle de preguntas de elección múltiple>>
<<Formato Moodle de preguntas de elección múltiple para el Profe>>
<<Formato Moodle de preguntas de elección múltiple con LastChoice>>

# Bloques Cloze (Moodle)

<<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. Helpers: identificadores y codificación de caracteres

### 2.1.1. Función `_sanitize_name`

Al construir un banco de preguntas es natural dar a cada grupo un nombre descriptivo que incluya el tema o el número de lección, como `L-07:Multicolinealidad` o `Tema3:Inferencia`. Estos nombres terminan incrustados en dos lugares problemáticos: el entorno `\element{nombre}{...}` de AMC y el entorno `\begin{cloze}{nombre-etiqueta}` de Moodle, y además en las reglas del Makefile que compila los ficheros. El carácter `:` es interpretado como delimitador de objetivo por `make` y como separador de etiqueta por algunos parsers  $\text{\LaTeX}$ , con resultados impredecibles: errores en la compilación, ficheros con nombres incorrectos o incluso silencio (el parser simplemente ignora parte del nombre).

Para que el usuario no tenga que preocuparse por estas restricciones —y pueda nombrar sus bancos con la notación que le resulte más natural—, todas las funciones de exportación llaman internamente a `_sanitize_name` antes de insertar el nombre en el código  $\text{\LaTeX}$ . El usuario escribe `L-07:Multicolinealidad`; la función entrega al compilador `L-07-Multicolinealidad`:

---

```
----- Helper _sanitize_name -----
def _sanitize_name(nombre):
    """Sustituye ':' por '-' en identificadores LaTeX (make y algunos parsers LaTeX lo rechazan)."""
    return nombre.replace(':', '-')
```

---

### 2.1.2. Función `codchar`

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("è", "\'~{e}")
    s = s.replace("í", "\'~{i}")
    s = s.replace("ó", "\'~{o}")
    s = s.replace("ü", "\'~{u}")
    s = s.replace("ñ", "\'~{n}")
    return s
```

---

## 2.2. Escritura de las cuestiones dentro del entorno `choices`

### 2.2.1. 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 desempquetado 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'
```

---

### 2.2.2. 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'
```

---

## 2.3. 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**.

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{}`.

---

Registro del motivo de rechazo de la cuestión

---

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

---

## Capítulo 3

# Escritura del código $\text{\LaTeX}$ para las preguntas en formato AMC

### 3.1. Función unificada `AMCblock`

Al desarrollar el módulo en sus primeras versiones, cada combinación de características de exportación daba lugar a una función independiente: `AMC`, `AMClastCh`, `AMCmc`, `AMClastChmc`, `AMCProfe`, `AMCmcProfe`... El problema es que la combinatoria crece rápidamente: basta añadir un nuevo parámetro para que el número de funciones necesarias se duplique. La solución es la habitual en diseño de APIs: colapsar toda la familia en una única función con parámetros opcionales que, por defecto, producen el comportamiento más común.

`AMCblock` es esa función unificada. Los tres ejes de variación que antes requerían funciones distintas se convierten ahora en argumentos de palabra clave con valores predeterminados:

- ¿Queremos opción comodín al final? → `last_choice=True` (por defecto `False`).
- ¿Queremos las opciones distribuidas en varias columnas? → `cols=2` (por defecto 1).
- ¿Queremos el diagnóstico de cuestiones descartadas por precondición? → `profe=True` (por defecto `False`).

Adicionalmente, `AMCblock` realiza dos transformaciones automáticas que antes había que gestionar a mano:

1. **Sanitización del nombre:** llama internamente a `_sanitize_name` para que los identificadores con `:` no provoquen errores en `make` ni en el compilador de  $\text{\LaTeX}$ .
2. **Conversión de matemáticas en modo display:** los bloques `$$...$$` del enunciado se reescriben como `\[...]`. AMC no reconoce el entorno `$$` de  $\text{\LaTeX}$  y los compilaría sin añadir el salto de línea esperado; el entorno `\[...]` es su equivalente correcto para matemáticas en bloque independiente.

Los parámetros que acepta son:

**nombre** Cadena que identifica la categoría del banco de ítems. Se sanitiza automáticamente (`:` → `-`) para compatibilidad con `make` y parsers  $\text{\LaTeX}$ .

**etiqueta** Identificador único de la variante concreta (el índice secuencial del iterador).

**enunciado** El bloque de texto principal. Los bloques `$$...$$` se convierten automáticamente a `\[...]`.

**cuestiones** Lista de tuplas (`texto`, `correcto`, `activa`[, `exp`]), tal como la produce el iterador de `ProblemaTipo`.

**last\_choice** Si `True`, añade `\lastchoices` con la opción comodín al final (ver sección anterior).

**cols** Número de columnas (`>1` activa `\begin{multicols}{N}\AMCBoxedAnswers`).

**profe** Si `True`, añade `\explain` con las cuestiones descartadas por precondición (ver sección anterior).

opc [instrucciones\_extra, texto\_lastchoice].

---

```
def AMCblock(nombre, etiqueta, enunciado, cuestiones,
             last_choice=False, cols=1, profe=False,
             opc=["", "Ninguna de las anteriores"]):
    """Genera el bloque AMC para una pregunta de tipo multichoice.

    Parámetros
    -----
    nombre      : identificador del grupo AMC (se sanitiza: ':' + '-')
    etiqueta    : etiqueta de la variante
    enunciado   : texto del enunciado (~`$$...$$` se convierte a `\\[...\\]`)
    cuestiones  : lista de (texto, correcto, activa[, exp])
    last_choice : si True, añade opción comodín con `\\lastchoices`
    cols       : número de columnas multicol (1 = sin multicol)
    profe      : si True, añade `\\explain` con cuestiones rechazadas
    opc        : [instrucciones_extra, texto_lastchoice]
    """
    InstruccionesAux = opc[0]
    OpcPorDefecto    = opc[1]
    nombre_tex      = _sanitize_name(nombre)
    enunciado_tex   = _re.sub(r'\$\$(.+?)\$', r'\\[1\\]', enunciado, flags=_re.DOTALL)

    ex = ''
    s = '\\element{' + nombre_tex + '}' + InstruccionesAux + '\\n'
    s += '\\begin{questionmult}' + nombre_tex + '-' + str(etiqueta) + '\\n'
    s += ' ' + enunciado_tex + '\\n'

    if cols > 1:
        s += '\\begin{multicols}' + str(cols) + '\\AMCBoxedAnswers\\n'
        ch_indent = ' '
    else:
        ch_indent = ''

    s += ch_indent + '\\begin{choices}\\n'
    for c in cuestiones:
        if c[2]:
            s += ' ' * 7 + ('\\correctchoice{' if c[1] else '\\wrongchoice {'} + c[0] + '\\n'
            elif profe:
                ex += ' cuestion: ' + c[0] + '; ' + str(c[1]) + '\\n'

    if last_choice:
        s += ' ' * 7 + '\\lastchoices\\n'
        s += ' ' * 7 + ('\\wrongchoice {' if any(c[1] for c in cuestiones) else '\\correctchoice{' + OpcPorDefecto + '\\n'

    s += ch_indent + '\\end{choices}\\n'

    if cols > 1:
        s += '\\end{multicols}\\n'

    if profe and ex:
        s += '\\explain{' + ex + '\\n'

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

---

## 3.2. 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:

---

```
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

```

---

### 3.2.1. 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): # Cuatro variantes
        var = (next(GenVar))
        f.write(AMC_VF(nombre, var[0], var[1], var[2]))

```

---

### 3.3. Ejemplo de uso de AMCblock

A continuación se presenta un ejemplo completo de diseño de una pregunta combinatoria. El argumento `last_choice=True` añade automáticamente la opción comodín al final y `cols=2` distribuye las respuestas en dos columnas:

---

```

from qbank import *
p = ProblemaTipo([
    "Considera ",
    [Supuesto("A ", v("A")), Supuesto("B ", v("B"))],
    [Supuesto("y A C. ", v("A") >> v("C")),
    Supuesto("y B D. ", v("B") ** v("D"))],
    "Indica qué es cierto: ",
    [Cuestion("C es verdadero", v("C")),
    Cuestion("D es verdadero", v("D"))],
], export={"last_choice": True, "cols": 2})

nombre = "MiPregunta"
with open(nombre + ".tex", "w") as f:
    for etiqueta, partes in p.por_partes():
        enunciado, cuestiones = partes[0]
        f.write(AMCblock(nombre, etiqueta, enunciado, cuestiones,
            last_choice=True, cols=2))

```

---

El campo `export` del `ProblemaTipo` permite guardar la configuración de exportación junto al problema en JSON (`last_choice`, `cols`), de modo que los scripts de generación puedan leerlo sin hardcodear los parámetros.

## 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).

La plantilla se almacena en la constante `_MOODLE_HEADER` y la función `_moodle_header` la instancia con el nombre del quiz y el preámbulo  $\text{\LaTeX}$  opcional:

---

```
_____ _MOODLE_HEADER y _moodle_header _____
_MOODLE_HEADER = """\
\\documentclass[11pt]{article}

\\usepackage[cm,headings]{fullpage}

\\usepackage{moodle}

\\usepackage{graphicx}

\\usepackage{fancyvrb}

\\ifPDFTeX                % FOR LATEX and PDFLATEX
  \\usepackage[utf8]{inputenc} % necessary
  \\usepackage[OT1]{fontenc}  % necessary
\\else                    % assuming XELATEX or LUALATEX
  \\usepackage{fontspec}
\\fi

{auxLaTeX}
\\newcommand\\peque{}

\\begin{document}

\\begin{quiz}{{nombre_tex}}

"""

def _moodle_header(nombre, auxLaTeX=""):
    return _MOODLE_HEADER.format(
        auxLaTeX=auxLaTeX,
```

```
) nombre_tex=_sanitize_name(nombre),
```

---

## 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 formateo 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 formateo 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 precondición (`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  $\text{\LaTeX}$  correspondientes a una pregunta de opción múltiple dentro del entorno `{multi}`.

---

```
_____ 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
```

---

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

---

<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"*.

---

```

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 y QuizVFMoodle

Estas funciones de alto nivel abren el flujo del archivo con `with open()`, vuelcan el preámbulo generado por `_moodle_header`, iteran sobre el diccionario de problemas inyectando las variantes y cierran el entorno principal `quiz`.

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}

```

---

### 4.5.1. QuizMoodle — función unificada

`QuizMoodle` replica, en el contexto de Moodle, la misma estrategia de unificación que motivó `AMCBlock` en `AMC`. Antes de esta versión, la opción de añadir la cuestión comodín «*Las demás opciones son falsas*» requería una función completamente separada (`QuizMoodleLastCh`). Ahora, un único parámetro booleano controla ese comportamiento:

- `last_choice=False` (valor por defecto) → genera preguntas sin opción comodín. Es el modo adecuado cuando la estructura del banco garantiza que siempre habrá al menos una respuesta correcta en cada variante.
- `last_choice=True` → inserta «*Las demás opciones son falsas*» como opción adicional, correcta si ninguna otra lo es y falsa en caso contrario. Es el modo habitual cuando el banco incluye variantes en que todas las respuestas podrían ser simultáneamente falsas.<sup>2</sup>

Internamente, la selección se implementa mediante una función de orden superior: la variable `cuerpo` apunta a `_ClozeMultiLastCh` o a `_ClozeMulti` según el valor de `last_choice`, y esa función se pasa como argumento a `_ClozeBlock` en el bucle de generación. Este patrón —almacenar en una variable la función que se quiere aplicar y pasarla como dato— evita un `if/else` repetido dentro del bucle de iteración y es una de las formas más limpias de escribir código condicional en Python.

### 4.5.2. QuizMoodle — función principal

---

```

Quiz para Moodle
def QuizMoodle(nombre, directorio, problema, last_choice=False,
               opc=["", "Las demás opciones son falsas"]):
    """Exporta un problema (o dict de problemas) al formato Moodle XML vía LaTeX.

    Parámetros
    -----
    nombre       : nombre del quiz (identifica el grupo en Moodle)
    directorio   : ruta al directorio de salida (con '/' al final)
    problema     : ProblemaTipo o dict {nombre: ProblemaTipo}
    last_choice  : si True, añade la opción comodín «las demás son falsas»
    opc         : [auxLaTeX, texto_lastchoice]
    """
    def creaDiccionario(x, key='key'):
        return x if isinstance(x, dict) else {key: x}
    problema = creaDiccionario(problema, nombre)
    opcPorDefecto = opc[1]

```

---

<sup>2</sup>En Moodle, a diferencia de AMC, no existe un comando `\lastchoices` que fije la posición de esa opción al barajar. Por eso el texto recomendado es «*Las demás opciones son falsas*» en lugar de «*Ninguna de las anteriores*»: el primero es semánticamente correcto en cualquier posición de la lista.

```

cuerpo = (lambda c: _ClozeMultiLastCh(c, OpcPorDefecto)) if last_choice else _ClozeMulti

with open(directorio + nombre + ".tex", "w") as f:
    f.write(_moodle_header(nombre, auxLaTeX=opc[0]))
    for i, nom in enumerate(problema):
        for etiqueta, partes in problema[nom].por_partes():
            f.write(_ClozeBlock(nom, etiqueta, partes, cuerpo))
    f.write("\end{quiz}\n\n\end{document}\n")

```

### 4.5.3. QuizMoodleProfe — versión docente

Genera el banco para revisión del profesor, mostrando el reparto de puntos por cuestión:

---

```

Quiz para Moodle versión Profe
def QuizMoodleProfe(nombre, directorio, problema, opc=["", ""]):
    def creaDiccionario(x, key='key'):
        return x if isinstance(x, dict) else {key: x}
    problema = creaDiccionario(problema, nombre)
    with open(directorio + nombre + ".tex", "w") as f:
        f.write(_moodle_header(nombre, auxLaTeX=opc[0]))
        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")

```

---

### 4.5.4. QuizVFMoodle

Para los bancos de preguntas 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]
    with open(directorio + nombre + ".tex", "w") as f:
        f.write(_moodle_header(nombre, auxLaTeX=auxLaTeX))
        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 QuizMoodle

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([
    "Considerare ",
    [
        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 $\mathcal{B} \rightarrow \mathcal{D}$ ", v("B") ** v("D")),
    ],
    "Indique qué opción es correcta: ",
    [
        Question("Entonces $\mathcal{C}$ es verdadero", v("C")),
        Question("Entonces $\mathcal{D}$ es verdadero", v("D")),
        Question("Entonces $\mathcal{E}$ es verdadero", v("E"), v("D")),
    ],
])
nombre = "EjemploMoodle"
directorio = "../ejemplos/"
QuizMoodle(nombre, directorio, p, last_choice=True)

```

---



---

```

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).

---

```
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 `AMCblock`:

---

```
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}`.

El identificador del entorno cloze se sanitiza mediante `_sanitize_name` (sustituye `:` por `-`) y se codifica con `codchar` para compatibilidad con L<sup>A</sup>T<sub>E</sub>X sin Unicode nativo.

---

```

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."""
  nombre_tex = codchar(_sanitize_name(nombre))
  s = " \\begin{cloze}{ " + nombre_tex + "-" + str(etiqueta) + " }\n"
  for enunciado, cuestiones in partes:
    s += " " + codchar(enunciado) + "\n"
    s += cuerpo(cuestiones)
  s += " \\end{cloze}\n\n"
  return s

```

---