Istruzioni per l’uso

Argomenti programma:

**--xml nome file (default='demo.xml') 🡪output moodle xml file**

**--questionFile nome file (default='domande.txt') 🡪 input txt question file**

**--footer nome file (default='footer.txt') 🡪 footer txt for questions**

**--workDir dir name (default='') 🡪 working dir**

**--category name (default=’** **"Default per 14BHDLZ\_16"’) 🡪 nome della categoria delle domande (serve per includerle in sottocategorie nel deposito domnade e trovarle più facilmente. Se non si scrive niente, tutte le domande vanno nella categoriale principale, quella di default)**

**--enableACE (default=False) 🡪 Enabling ACE editor**

**--defaultPythonGrade (default=26.0)🡪 Python grade**

**--defaultEssayGrade (default=2.0) 🡪 Essay grade**

Il programma legge le domande da un file di testo (nota: deve essere codificato in utf-8 per evitare problemi con le lettere accentate), che ha un formato proprio e che permette di inserire tutti i campi delle domande (attualmente solo quelle di Coderunner, inclusi eventuali file che devono essere letti durante l’esercizio, e gli essay).

**Domande Coderunner**

Il testo della domanda viene completato dal footer, che è uguale per tutte le domande, e che contiene le indicazioni su come usare l’interfaccia e il link alla documentazione Python.

All’interno del file possono essere presenti più di una domanda, e il file xml finale contiene l’elenco di tutte le domande convertite in formato xml moodle che quindi si possono importare direttamente all’interno di exercise o exam.

I vari file possono essere memorizzati in una directory di lavoro (che si può specificare al programma), in cui verranno poi anche salvati gli output prodotti. Nella directory del sorgete deve SEMPRE essere incluso il file template.py che contiene il codice necessario a creare la stringa json da inviare al server che fa girare pyhon (e per recuperare e leggere correttamente l’output dell’esecuzione). Quindi riassumendo:

* Directory del sorgente: deve includere anche i file template.py e footer.txt
* Directory di lavoro: contiene il file con le domande e tutti gli eventuali file (da leggere…) che devono essere inclusi nelle domande

**Essay**

Gli essay, servono per le domande di teoria, e consistono in un testo (codificato in markdown), e un eventuale template di risposta (sempre in markdown)

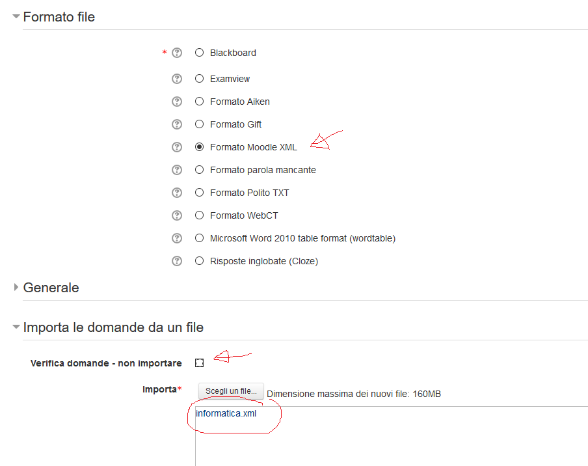
## Upload in moodle

Una volta creato il file xml, i passaggi per importarlo in exercise/exam sono i seguenti:

* Deposito domande 🡪 importa



* Selezionare come formato moodle xml, togliere il check su verifica domande e trascinare il file xml generato



## Formato file di testo delle domande coderunner

**QUESTION**

Identificatore della domanda (campo obbligatorio)

È la stringa che nel deposito domande di exercise permette di distinguere una domanda dall’altra. Questa stringa NON compare nel testo della domanda

**TEXT**

Testo della domanda (campo obbligatorio). All’interno di moodle il testo deve essere in HTML. Per semplificare la vita, il testo della domanda deve essere scritto in markdown e viene convertito automaticamente in HTML (il markdown è decisamente più semplice da gestire che mettersi a scrivere tag HTML ovunque)

Link a definizione formato markdown:

<https://daringfireball.net/projects/markdown/syntax>

NOTA: per inserire codice nel markdown indentate ogni linea di codice con un tab. NON usate ‘’’ per delimitare un blocco di codice, dal momento che moodle poi non prende correttamente gli a capo (li considera come spazi).

NOTA2: i backtick (‘) in un testo normale creano un blocco a parte che interrompe la linea (questo per come funziona moodle. Il tag <code> che viene generato, in moodle crea un box a parte su una linea separata interrompendo il testo)

NOTA3: nel markdown si può inserire codice HTML, che viene riportato as is. Per lo stesso motivo, non usate <> per definire parti del testo (ad esempio per identificare i campi di un file) perché poi vengono interpretati come tag HTML (e quindi non compaiono nel testo)

**ANSWER**

Questo campo (facoltativo) contiene tutto quello che si vuole che compaia come indicazione/suggerimento nel testo della risposta. È un campo plain text perché deve essere codice Python. Si può anche non mettere niente

**FILES**

Nel caso l’esercizio preveda la lettura di file, si può fare in modo che i file siano già inglobati nella domanda stessa (anche se non vengono visualizzati dal momento che il “file browser” di coderunner non si può usare dal momento che la nostra versione di moodle non è aggiornata, e non si riesce ad aggiornare per tempo).

Si possono aggiungere tutti i file che si vogliono. In questo caso, basta elencare il nome (uno per riga). I file veri e propri si devono trovare nella working dir, devono essere file testuali (al momento…) e possono avere suffisso qualsiasi.

NOTA: nel caso ci siano dei file in lettura, ho aggiunto in automatico al campo answer delle istruzioni automatiche di stampa del contenuto del file del tipo:

**print(open(filename,‘r’).read())**

Il campo è facoltativo.

## Formato file di testo delle domande di tipo essay

**ESSAY**

Identificatore della domanda (campo obbligatorio)

È la stringa che nel deposito domande di exercise permette di distinguere una domanda dall’altra. Questa stringa NON compare nel testo della domanda

**TEXT**

Testo della domanda (campo obbligatorio). Come nel caso precedente, anche qui va scritto in markdown (con le regole specificate sopra)

**ANSWER**

Questo campo (facoltativo) contiene tutto quello che si vuole che compaia come indicazione/suggerimento nel testo della risposta. Al contrario delle domande per coderunner, questo è un campo che va scritto in markdown (cosa che consente di avere più flessibilità nel formato)

## Esempio file di domande

**ESSAY**

**Somma ca2**

**TEXT**

**Qual è il risultato della seguente somma, nella rappresentazione in complemento a 2 su 8 bit?**

**10111110 + 01110011**

**ANSWER**

**Risultato:**

**QUESTION**

**Incroci**

**TEXT**

**# Incroci**

**Si scriva un programma per la gestione delle precedenze ad un incrocio di tre strade denominate N, S, E. Si scriva un programma per la gestione delle precedenze ad un incrocio di tre strade. Il programma deve leggere gli arrivi sulle varie file tramite un file che si chiama “arrivi.txt”. Ogni riga di tale file rappresenta un certo istante di tempo in questo modo:**

**TARGA\_N TARGA\_S TARGA\_E**

**Dove TARGA è una stringa di 7 caratteri, TARGA\_N indica un arrivo dalla strada N, TARGA\_S indica un arrivo dalla strada S, TARGA\_E indica arrivo dalla strada E. Nel caso in cui non ci sia una macchina proveniente da una determinata strada TARGA sarà rappresentata dalla stringa \*\*XXXXXX\*\*.**

**Per ogni istante di tempo considerato (ovvero per ogni riga del file in ingresso) il programma deve scrivere sul file di output “precedenze.txt”, la macchina che può attraversare l’incrocio secondo la seguente regola: ha precedenza la macchina della fila più lunga. Nel caso di file di ugual lunghezza si scelga la prima esaminata (quindi, l’ordine di precedenza è N->S->E). Terminata la lettura del file in input continuare il programma deve continuare a scrivere in ordine le targhe delle macchine che attraversano l’incrocio fino all’esaurimento delle stesse.**

**Una macchina che non passa ad un certo istante di tempo all’incrocio rimane in fila. I nuovi arrivi di macchine ad ogni istante di tempo si accodano alle rispettive file.**

**# Esempio file arrivi.txt (in input)**

**AA100BB CC200FF GG300HH**

**CC100ZZ YY200PP VV500KK**

**NN200KK XXXXXXX HH200DD**

**XXXXXXX FF600PP XXXXXXX**

**# Esempio esecuzione:**

**c:> python incrocio.py**

**File precedenze.txt (creato dal programma)**

**AA100BB**

**CC200FF**

**GG300HH**

**CC100ZZ**

**YY200PP**

**VV500KK**

**NN200KK**

**FF600PP**

**HH200DD**

**ANSWER**

**# this is an example of answer filling**

**# i file specificati NON sono visibili direttamente, ma esistono e si possono aprire**

**FILES**

**arrivi.txt**