Sonic TBL: Un Percorso Sonico da Creatività a Didattica dell'Informatica

Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro

DIBRIS, Università degli Studi di Genova, Italy
{name.surname}@unige.it

Sommario

Nel presente articolo vengono presentate alcune attività basate su Sonic Pi, nato per live music coding performance, e sul metodo didattico Team-Based Learning (TBL) per condurre esperimenti didattici sull'introduzione di concorrenza e multithreading in vari momenti del percorso della Laurea Triennale in Informatica della nostra Università.

1 Contesto e Motivazioni

Imparare la programmazione è considerato universalmente un processo di apprendimento difficile e faticoso in quanto strettamente legato alle capacità di ragionamento logico, problem solving e pensiero creativo. Le numerose iniziative nazionali ed internazionali volte ad anticipare l'introduzione dei concetti computazionali nella scuola primaria e secondaria sono un chiaro esempio del tentativi in atto per migliorare la preparazione di base della disciplina. Nonostante questi sforzi e nonostante l'enorme gap con richiesta del mercato professionale, i corsi di laurea in Informatica soffrono di un elevato tasso di abbandono nei primi anni. Da recenti statistiche relative ai corsi di laurea italiani, risulta che uno studente su quattro iscritto ad Informatica passa ad un'altro corso di laurea o lascia l'università tra il primo e secondo anno. Una delle cause di questo alto tasso di abbandoni è sicuramente legata alle difficoltà iniziali nell'affrontare corsi di programmazione offerti a livello universitario. Per questo motivo la necessità di proporre metodi didattici innovativi, ad esempio basato sull'individuazione di Notional Machine adatte al modello mentale di studenti provenienti da diversi tipi di scuole, è diventato un obiettivo della maggior parte dei corsi di laurea italiani.

Il corso di laurea in Informatica della nostra Università attira negli ultimi anni più di 300 matricole con background eterogeneo. Gli studenti sono motivati principalmente dalle opportunità di lavoro offerte dalla laurea, e spesso solo una piccola parte di loro ha motivazioni e attitudini favorevoli all'apprendimento. Le lacune nelle conoscenze e nelle competenze di base sono spesso complementari. Ad esempio, chi ha un background di programmazione di solito non ha una solida base matematica e viceversa. Gli studenti con poche competenze di base possono quindi essere scoraggiati dall'osservare compagni di corso che raggiungono gli obiettivi dei corsi introduttivi con meno sforzo. Gli studenti con un background adeguato, invece, spesso non ricevono sufficienti incentivi per approfondire gli argomenti forzatamente introduttivi. Molti dei nostri studenti inoltre hanno difficoltà nel lavorare in gruppo con carenze generali nel relazionarsi a compagni e docenti. Il problema persiste anche negli anni successivi, in cui molti studenti sembrano non aver acquisito né metodologie adeguate né la necessaria autonomia. A partire dall'anno accademico 2019/20 il corso di laurea ha avviato una serie di azioni di innovazione didattica e alcuni esperimenti per introdurre metodi didattici attivi strettamente integrati con i corsi dei primi anni. In questo contesto sono stati introdotti metodi didattici non tradizionali come flipped class, team-based learning, think-pair-share, debate e argomenti aggiuntivi ai contenuti tradizionali dei corsi per creare collegamenti, stimoli, puntatori ad argomenti avanzati

e gruppi di lavoro che potessero coinvolgere studenti di diversi anni (es studenti di dottorato e magistrale come mentor per studenti della triennale).

Nel presente articolo, esaminiamo una serie di attività basate sull'uso combinato di un domain specific language chiamato Sonic Pi[1], nato per live music coding performance, e del metodo didattico Team-Based Learning (TBL) per condurre esperimenti didattici sull'introduzione di concorrenza e multithreading in vari momenti del percorso della laurea in Informatica.

Sonic Pi è un domain-specific language basato su un server Supecollider e supportato da un editor di facile utilizzo per la creazione di basi musicali. Il sistema permette di introdurre sia concetti computazionali di base (strutture dati, costrutti, funzioni, ecc) sia concetti che giocano un ruolo sempre più importante nella programmazione come multi-threading e hot code swap [1, 2, 10, 3]. La semplicità dello strumento fornisce un'esperienza di apprendimento naturale e coinvolgente, in particolare attraverso il live coding (basato appunto su hot code swapping) e la ricca resa sonora (multi-strumentale grazie all'uso di sample di sintetizzatori) dei corrisponenti programmi. Il nostro esperimento didattico è stato ripetuto in tre diversi anni accademici, pre e post pandemia, esplorando formati e programmando interventi sia al primo che al terzo anno, anche per valutare, in momenti diversi del percorso di studio, la percezione degli studenti verso questo tipo di attività. In questo articolo presenteremo brevemente le varie edizioni, esplorando obiettivi di apprendimento, risultati preliminari ed implicazioni degli esperimenti facendo riferimento a quattro insegnamenti: Introduzione alla Programmazione (IP) primo semestre del primo anno, Architettura dei Calcolatori (ADC) annuale al primo anno, Programmazione Concorrente e Algoritmi Distribuiti (PCAD) primo semestre del terzo anno, Informatica per Creatività, Didattica e Divulgazione (ICDD) insegnamento a scelta al terzo anno.

2 Fase 1: Programmazione Creativa (2019/20)

Durante l'anno accademico 2019/2020, abbiamo avviato un progetto pilota che ha utilizzato Sonic Pi in combinazione con la metodologia didattica del Team-Based Learning (TBL). Questo progetto era rivolto alle matricole del primo anno iscritte al corso di laurea in informatica. La sperimentazione si è posta due obiettivi principali. Da un lato, volevamo fornire agli studenti competenze trasversali altamente richieste nel mondo del lavoro, come la creatività, la capacità di lavorare in squadra e di comunicare efficacemente con gli altri, nonché la capacità di mediare le proprie idee con quelle degli altri. L'intento era abbattere lo stereotipo dell'informatico come persona asociale e favorire il lavoro in un ambiente interdisciplinare, collaborando con persone provenienti da diverse esperienze professionali. Per raggiungere questo obiettivo, abbiamo creato gruppi di lavoro omogeneamente eterogenei, utilizzando variabili come la scuola di provenienza, il genere, il voto di maturità, eventuali esperienze lavorative e altri fattori. Il secondo obiettivo era legato alla motivazione all'apprendimento degli studenti, al fine di contrastare l'alto tasso di abbandono accademico. Il corso di introduzione alla programmazione è un corso semestrale incentrato sulla programmazione imperativa in C++. Tuttavia, per gli studenti provenienti da istituti scolastici in cui l'informatica è materia curricolare, il corso può offrire inizialmente pochi stimoli e sembrare limitato nella sua visione della programmazione. D'altra parte, per gli studenti provenienti da percorsi scolastici che non prevedono lo studio dell'informatica (come ad esempio un liceo scientifico tradizionale), il corso può risultare più difficile e, di conseguenza, demotivante poiché i colleghi con un background informatico riescono a raggiungere gli stessi risultati con meno tempo e fatica. A tal fine, abbiamo progettato

[1] https://sonic-pi.net

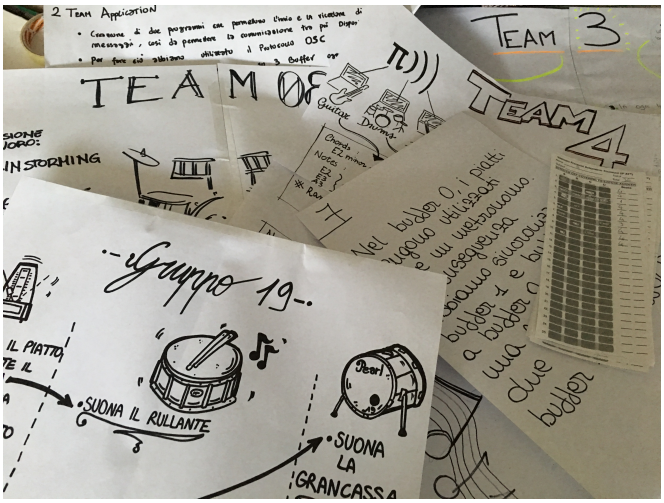

Figura 1: Creatività e lavoro di grupppo a servizio dell'apprendimento della programmazione.

un'attività basata sull'introduzione di concetti avanzati di programmazione come concorrenza, temporizzazione e sincronizzazione in un loro contesto naturale di applicazione come la creazione di un brano musicale multistrumentale. L'analisi dell'esperienza e dei dati raccolti è discussa in dettaglio in [14].

La Fig. 6 in Appendice mostra un esempio degli esercizi di warm-up su Sonic Pi proposti in questa attività.

3 Fase 2: Knowledge Transfer (2021/22)

Basandoci sull'esperienza acquisita nel primo anno dell'esperimento, abbiamo ulteriormente perfezionato e adattato il formato didattico nell'anno accademico 2021/2022 per gli studenti del terzo anno iscritti ai corsi di PCAD e ICDD. Questi corsi hanno fornito una piattaforma ideale per esplorare la concorrenza e la sua applicazione in contesti creativi. L'obiettivo principale di apprendimento era introdurre gli studenti al concetto di multithreading in Python. Tuttavia, abbiamo affrontato questo obiettivo partendo dalla concorrenza in Sonic Pi e poi passando a Python utilizzando la libreria `python-sonic`. Questa interfaccia ha permesso agli studenti di combinare le capacità di multithreading di Python con le funzionalità di Sonic Pi all'interno dell'ambiente di sviluppo Python. La libreria consente specificamente di incorporare il codice musicale di Sonic Pi all'interno dei thread di Python. Ciò permette una transizione graduale da un dominio all'altro, permettendo agli studenti di tradurre la logica del programma concorrente da Sonic Pi a Python utilizzando la libreria `threading`. Per gli studenti iscritti al corso ICDD, l'esperimento ha permesso anche di esaminare l'attività dal punto di vista dell'instructional design, analizzando gli elementi pedagogici dell'esperimento stesso, come gli obiettivi di apprendimento, la scelta del linguaggio di programmazione per la metodologia didattica, ecc. In queste due edizioni delle attività l'enfasi si è quindi spostata su concorrenza e multithreading mantenendo sempre un forte contatto sul dominio specifico degli strumenti adottati (live music coding) e gli argomenti dei corsi.

4 Fase 3: Misconcezioni nella Concorrenza (2022/23)

Per l'anno accademico 2022/23, ci siamo concentrati sull'obiettivo di introdurre gli studenti al concetto di concorrenza attraverso una serie di brevi compiti pratici, ognuno progettato per affrontare specifiche misconcezioni sulla concorrenza, vedi ad esempio [9, 12, 13], e concetti di base della programmazione. In questo modo, viene consentito agli studenti di costruire gradualmente la loro comprensione affrontando ad esempio le diverse dimensioni dello scope delle variabili in presenza dei thread (globali, automatiche, thread-local), chiamate di funzioni e passaggio di parametri in presenza di thread, sincronizzazione e race condition, correttezza dei programmi multithreaded. Alcuni task si sono concentrati sulla comprensione del programma, mentre altri richiedevano la scrittura del codice. I task di comprensione [8] rappresentano un approccio costruttivista all'insegnamento della programmazione, in cui lo studente interagisce con un artefatto che rappresenta il programma, ad esempio un pezzo di codice. Attraverso questa interazione, lo studente viene stimolato a costruire e affinare il proprio modello mentale della macchina nozionale (*notional machine)* sottostante [4, 7, 11]. La Figura 2 mostra un esempio di task, incluso nella Team App di TBL, collegato a riformulazioni in Sonic Pi di misconcezioni sulla concorrenza con attenzione su scope delle dichiarazioni (es. globale, automatico, thread local) e data race. La Figura 3 mostra invece un esempio di task dove il suono viene usato per "sentire" (es. differenza tra suonare un accordo o una sequenza di note) possibili anomalie legate all'esecuzione out of order di un programma.

Un modulo con aspetti di base è stato proposto al termine del corso ADC nel contesto di alcune ore dedicate alle architetture multicore e alle GPU. Un modulo con task avanzati (es. strutture dati concorrenti e sincronizzazione) è stato proposto a PCAD anche per valutare la diversa percezione di studenti che avevano già svolto attività con Sonic Pi maggiormente orientate all'uso creativo della programmazione. L'analisi dell'esperienza e dei dati raccolti è discussa nel rapporto tecnico [6].

5 Discussione

La sperimentazione sul campo nel corso delle diverse edizioni dell'attività ha permesso di sintetizzare un formato, ribattezzato Sonic TBL [6], che combina le caratteristiche migliori del metodo TBL e di strumenti come Sonic Pi. Riteniamo che, a seguito dei diversi raffinamenti descritti in questo articolo, il formato abbia raggiunto nel corrente anno accademico forma e sostanza adeguati al livello universitario come testimoniato da una vasta partecipazione, circa 200 studenti, alle attività proposte durante il corso dell'anno accademico. Dai dati collezionati nell'anno accademico 2022/23, vedi [5], i risultati ottenuti nei vari task enfatizzano la diffusa presenza di misconcenzioni tra gli studenti sia su concetti relativi a nozioni di base di concorrenza, sia su concetti generali di programmazione calati nel contesto della programmazione concorrente (es. scoping a livello di funzione e thread). In generale l'approccio combinato TBL e Sonic Pi consente di offrire agli studenti un'esperienza di apprendimento che integra attività pratiche di programmazione anche con aspetti avanzati rispetto al tipico programma del primo anno, applicazioni su un dominio concreto come la creazione di musica e apprendimento collaborativo. Inoltre il metodo ha permesso la graduale introduzione della programmazione multithreaded al termine del primo anno, un aspetto ormai fondamentale visto il sempre crescente interesse ad esempio per computazione ad alte prestazioni basata su multicore e GPU.

Per quanto riguarda il corso di Architetture dei Calcolatori del primo anno, 23 studenti su 130 hanno compilato il questionario finale. Nella domanda A, "Penso che Sonic Pi sia utile per capire la concorrenza", il valore mediano è risultato 4 su scala di Likert da 1 a 5. Per la

A

```
in_thread do
  use_synth :piano
  x = 40
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
 use_synth :kalimba
  x = 40
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

B

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

C

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

x = 60
in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

Voting cards: Which answer is true?

- In B and C, different threads operate on a common resource and the result depends on the order in which the different threads execute their instructions;
- In A and C, different threads operate on a common resource and the result depends on the order in which the different threads execute their instructions;
- In all three programs, there are no resources shared between threads;
- In A and C, there are no resources shared between threads. The result depends on the order in which the instructions of the different threads are executed.

Figura 2: Team Apps - Misconcezione concorrenza in Sonic Pi.

domanda B, “Penso che l’attività sia stata efficace per introdurre e approfondure concetti della programmazione concorrente”, il valore mediano è risultato 3 su scala di Likert. La Figura 4 mostra la distribuzione delle risposte ottenute. Il 74% dei partecipanti ha trovato l’attività di difficoltà adeguata, il 22% ha ritenuto l’attività difficile, e il rimanenre 4% facile. Il 22% degli studenti ha trovato il lavoro in team molto utile, il 26% si è dimostrato neutrale, il 13% lo ha ritenuto poco utile e il 9% per nulla utile.

Per quanto riguarda il corso di Programmazione Concorrente e Algoritmi Distribuiti del terzo anno, 16 studenti su 54 hanno compilato il questionario finale. Nella domanda A, “Penso che Sonic Pi sia utile per capire la concorrenza”, il valore mediano è risultato 4 su scala di Likert da 1 a 5. Per la domanda B, “Penso che l’attività sia stata efficace per introdurre e approfondure concetti della programmazione concorrente”, il valore mediano è risultato 4 su scala di Likert. La Figura 5 mostra la distribuzione delle risposte ottenute. L’80% dei partecipanti ha trovato l’attività di difficoltà adeguata, il 18% ha ritenuto l’attività difficile, e il rimanenre 4% facile. Il 25% degli studenti ha trovato il lavoro in team molto utile e il

```
use_synth :piano

note=[52,55,59,40]
i=0

define :foo do |x|
  in_thread do
     play note[x]
  end
end

3.times do
  foo i
  i+=1
end
```

Voting cards: Which answer is true?

(1) The program has no race conditions.
(2) The program creates only one thread.
(3) The program plays the notes in the "`note`" list in sequence.
(4) The program plays the E minor chord.

Gallery walk:
Describe in detail the behavior of the script with particular pay attention to the changes to the value of the variable "`i`" and to the possible sequences of notes play.

Figura 3: Team App - Sentire gli interleaving con Sonic Pi.

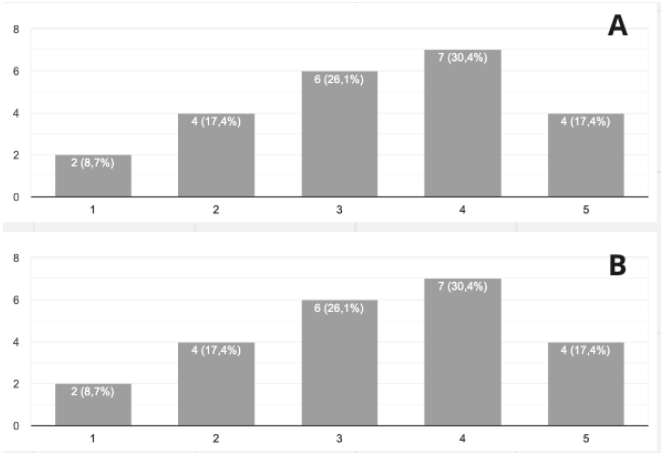

Figura 4: Distribuzione risposte al questionario finale.

75% sufficientemente utile. Da questi dati si può notare come affrontare le misconcezioni più comuni in aspetti di base e avanzati di programmazione sia ritenuto utile da studenti sia del primo anno che del terzo. Inoltre gli studenti del terzo anno, forse anche grazie all'esperienza acquisita affrontando progetti ed esami, tende a dare molto più valore al lavoro in team rispetto ai novizi.

Riferimenti bibliografici

[1] Samuel Aaron, Alan F. Blackwell, and Pamela Burnard. The development of sonic pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music Technology and Education*, 9(1):75–94, 3 2016.

[2] Samuel Aaron, Dominic A. Orchard, and Alan F. Blackwell. Temporal semantics for a live coding language. In Alex McLean, Michael Sperber, and Henrik Nilsson, editors, *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design, FARM@ICFP 2014, September 1-3, 2014*, pages 37–47, Gothenburg, Sweden, 2014. ACM.

[3] Roberto Agostini, Leo Izzo, and Giovanni Nulli. Spiegare il "caso" e l'array nel live coding musicale. In *Proc. Didamatica 2022*, page 410–419, Milan, Italy, 2022. AICA.

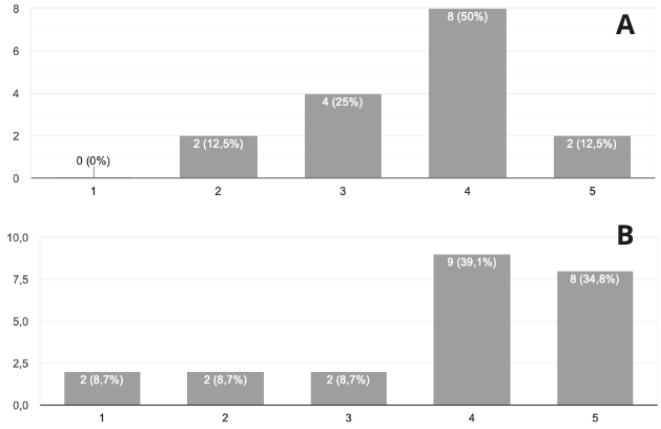

Figura 5: Distribuzione risposte al questionario finale.

[4] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[5] Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro. Concurrency education with sonic pi: Hear and play mistakes and misconceptions in multithreaded programs, technical report. Technical report, DIBRIS, Università deli Studi di Genova, 2023.

[6] Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro. Concurrency Education with Sonic Pi: "Hear" and "Play" Mistakes and Misconceptions in Multithreaded Programs. Under review, available on request. Technical report, DIBRIS, University of Genoa, July 2023.

[7] Benedict du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3):237–249, 1981.

[8] Cruz Izu, Carsten Schulte, Ashish Aggarwal, Quintin Cutts, Rodrigo Duran, Mirela Gutica, Birte Heinemann, Eileen Kraemer, Violetta Lonati, Claudio Mirolo, et al. Fostering program comprehension in novice programmers-learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19*, pages 27–52. ACM, Aberdeen, Scotland, UK, 2019.

[9] Jan Lönnberg. Student errors in concurrent programming assignments. In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Baltic Sea '06, page 145–146, New York, NY, USA, 2006. Association for Computing Machinery.

[10] Christopher Petrie. Programming music with sonic pi promotes positive attitudes for beginners. *Computers & Education*, 179:104409, 2022.

[11] Juha Sorva. Notional machines and introductory programming education. *ACM Trans. Comput. Educ.*, 13(2), jul 2013.

[12] Filip Strömbäck. *Teaching and Learning Concurrent Programming in the Shared Memory Model*. PhD thesis, Linköping University, Sweden, 2023.

[13] Filip Strömbäck, Linda Mannila, Mikael Asplund, and Mariam Kamkar. A student's view of concurrency - a study of common mistakes in introductory courses on concurrency. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 229–237, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Daniele Traversaro, Giovanna Guerrini, and Giorgio Delzanno. Sonic Pi for TBL Teaching Units in an Introductory Programming Course. In *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20 Adjunct, page 143–150, New York,

Warm-up exercise 1.

- Copy the following code to Sonic Pi and run:

```
sample :loop_amen
sample :loop_amen
```

- Add between the two instructions `sleep 0.87` then run.
- Loop the code exactly three times (a kind of C + for-loop)
- Turn it into an infinite loop.
- Add the following code in the same buffer:

```
loop do
  if one_in(2)
    sample :drum_heavy_kick
  else
    sample :drum_cymbal_closed
  end
  sleep 0.877
end
```

How can we play the loops simultaneously? (hint: use `in_thread`)

- Modify the code so that the probability of executing `sample: drum_heavy_kick` is lower than the current one.
- Insert a `sleep (0.3)` between the two threads. What happen? Synchronize the two threads (hints: use `cue / sync`).
- Given the following code (to be copied into a new Sonic Pi buffer):

```
live_loop :b do
 play :e4, release: 0.5
  sleep 0.5
end
live_loop :c do
 sample :bd_haus
  sleep 1
end
```

Synchronize the two live loops (hints: whenever a loop loops, it generates an event of type cue).

- In general we can also synchronize more than two threads. Run the following code: the master represents the orchestra director and the slaves the two musicians who must be synchronized with the director:

```
#master
in_thread do
  loop do
    cue :tick
    sleep 2
    cue :tock
    sleep 2
  end
end

  #slave1
  in_thread do
    loop do
      sync :tick
      sample :drum_cowbell
    end
  end
  #slave2
  in_thread do
    loop do
      sync :tock
      sample :drum_splash_soft
    end
  end
```

Warm-up exercise 2. Copy the following code into a new Sonic Pi buffer:

```
a = [6, 5, 4, 3, 2, 1]

live_loop :shuffled do
  a = a.shuffle
  sleep 0.5
end

live_loop :sorted do
  a = a.sort
  sleep 0.5
  puts "sorted: ", a
end
```

Where `a` is a Sonic Pi list, `shuffle` is the function that shuffles the Sonic Pi list, `sort` is the function that sorts the list in ascending order. Run the program. What do you observe? Do the two loops access the shared resource in mutual exclusion? Eliminate the race condition by using the Sonic Pi Time State (hint: use the `set` and `get` methods).

Warm-up exercise 3.
Given the following code:

```
loop do
 sample :perc_bell, rate: (rrand 0.125, 1.5)
 sleep rrand(0.2, 2)
end
```

Modify the program so that the `rate` time varies between 0.125 and 1.5, and the `sleep` time between 0.2 and 2 (hint: randomization). Run the program several times. Is the melody always the same? Why?

Prompt task
You are consulting an event manager who wants to organize a live music event where the band is completely simulated by Sonic Pi. His DJ is an expert in music, but not computer programming. How would you help him make music with Sonic Pi? Use your computer programming skills. Example and tips. Use 3 different buffers. In each buffer insert a live loop with sample commands for a particular drum element (cymbals, snare, bass). Then use the cue / sync commands to synchronize the live loops in the different buffers. Finally use sleep to enter the timing between the playback of the various samples. Use hi-hat (drum_cymbals) as a metronome. Then try to synchronize the buffers from different computers connected to the same network using the OSC protocol. Time: 1 hour and 45 minutes.

Figura 6: Esercizi di riscaldamento su Sonic Pi.

NY, USA, 2020. Association for Computing Machinery.