Imparare la chimica analitica con il coding in Python

Giovanni Merola, Jacopo Campagna, Samuele Celletti, Michele Maglia
Istituto Tecnico Industriale Statale "Stanislao Cannizzaro", Colleferro (RM)

e-mail: giovanni.merola@itiscannizzarocolleferro.it

Abstract. The innovation of teaching and learning methodologies in STEM (Science, Technology, Engineering, and Mathematics) is a priority for educational systems globally. It represents a fundamental challenge in schools to improve teaching effectiveness and the acquisition of technical, creative, digital, communication, collaboration, problem-solving, flexibility, adaptability, and critical thinking skills. Furthermore, the Ministry of Education aims to promote the establishment of laboratory spaces and the provision of suitable digital tools to support curriculum learning and the teaching of STEM subjects. The use of AI in STEM education opens new possibilities for engaging and interactive learning, fostering curiosity and exploration, and preparing students for the challenges of the digital age. This article presents a didactic approach to common problems in analytical chemistry, but also suitable for all STEM disciplines, which utilizes the foundations of the Python programming language to develop algorithms for solving various chemistry-related problems, including stoichiometry and analytical chemistry.
Moreover, this educational program, divided into multiple modules, consists of laboratory sessions conducted in a computer or multimedia classroom. During these sessions, students will learn how to set up calculations and begin programming while gradually familiarizing themselves with the Python program's commands and syntax. Among the codes developed so far by a group of students from ITIS Cannizzaro in Colleferro (Rome), there are a unit converter, a tool for calculating the pH of weak and strong acids and bases, a tool for calculating titration curves, and others currently under development such as Webapps. The utilization of AI in this approach brings numerous benefits, including personalized learning, adaptive feedback, data analysis for optimization, real-time assistance, and fostering curiosity and digital preparedness.

Keywords: coding; Python; pH; STEM; chimica analitica

1. Introduzione

L'apprendimento delle discipline STEM (Scienze, Tecnologia, Ingegneria e Matematica) rappresenta una priorità nei sistemi educativi globali. Nell'ambito scolastico, è fondamentale migliorare l'efficacia didattica e sviluppare competenze tecniche, creative, digitali, di comunicazione, collaborazione, problem solving, flessibilità, adattabilità al cambiamento e pensiero critico. Il Ministero dell'Istruzione ha promosso l'implementazione di spazi laboratoriali e l'utilizzo di strumenti digitali adeguati per sostenere l'insegnamento delle discipline STEM [1].

Il linguaggio Python è stato sviluppato dall'olandese Guido van Rossun negli anni Novanta del secolo scorso per migliorare il linguaggio Perl. La sintassi di Python è molto più semplice e questo ne migliora la leggibilità del codice sorgente. L'uso di Python si è diffuso rapidamente nella community e in poco tempo ha raggiunto la notorietà di altri linguaggi *web oriented* come Java e lo stesso Perl [2]. Nel corso del tempo il linguaggio di programmazione ha beneficiato di una vera e propria evoluzione e grazie al fatto d'essere open source, tutti hanno contribuito a migliorarlo. Attualmente Python è utilizzato da molte società di *software development* e persino da molti colossi del Web, come il motore di ricerca Google e la piattaforma di condivisione dei video online YouTube. L'origine del nome è davvero particolare. Il creatore del linguaggio è un appassionato del gruppo comico inglese Monty Python. Questa passione ha influito sulla scelta del nome da dare al linguaggio di programmazione che ha un'origine decisamente bizzarra [3].

Python è un linguaggio Open Source che si sta diffondendo rapidamente in numerosi ambiti, tra cui quello scientifico. Il suo utilizzo si sta consolidando grazie alle caratteristiche di semplicità, eleganza, modernità, integrazione ed estendibilità. Questo fa di Python un candidato ideale per applicazioni scientifiche quali analisi dati, simulazioni, grafici, manipolazione algebrica e così via, anche a discapito dei numerosi software ritenuti indispensabili per tali scopi. Python ha una curva di apprendimento più rapida rispetto a Java e C++. Questo è dovuto alla sua sintassi intuitiva e alla disponibilità di una vasta documentazione e risorse online. Anche i principianti possono iniziare a scrivere codice funzionante in poco tempo.

Numeric, Scientific, SciPy, ChemPy [4] sono alcuni dei moduli che permettono moltissime operazioni di calcolo scientifico con estrema semplicità, esistono librerie per il calcolo parallelo; infine esistono moduli specifici per innumerevoli applicazioni (fisica, astronomia, chimica, ...), facilmente e liberamente reperibili, installabili e ben documentati [5]. A tuttoggi quindi il softwtare in ambito scientifico riveste un ruolo chiave nella elaborazione, gestione e presentazione dei dati sia nella ricerca, ma anche in ambito didattico [6].

La comunità scientifica ha già riconosciuto l'importanza di utilizzare il coding in Python nell'insegnamento della chimica, per questo sono stati svilup-

pati diversi moduli, come ad esempio, ChemPy, che offre funzionalità chimiche utili implementate in Python. Inoltre, è stato evidenziato come il computer possa aiutare nell'equilibrio delle equazioni chimiche ed è stata pubblicata una serie di quaderni Jupyter che utilizzano Python per un corso di chimica analitica [5, 7].

Questo articolo propone un approccio didattico innovativo per l'insegnamento della chimica analitica, basato sull'utilizzo del coding in Python. Lo scopo principale è fornire agli studenti uno strumento pratico e interattivo per affrontare le sfide della chimica analitica, sviluppando competenze chiave nel contesto delle discipline STEM. L'approccio didattico di costruire algoritmi per risolvere problemi fisici o chimici è importante perché promuove il pensiero logico-analitico e rafforza la comprensione dei concetti fondamentali, consentendo agli studenti di applicare le conoscenze teoriche in contesti pratici. Inoltre, sviluppa le abilità di problem solving e favorisce l'integrazione tra teoria e pratica nelle discipline scientifiche. Appare chiaro che il pensiero computazionale venga oggi riconosciuto come una competenza fondamentale per avere successo nelle discipline STEM, ma anche negli altri ambiti disciplinari. Programmare è un potente strumento di pensiero, di espressione e di crescita personale perché, imparando come si programma, si impara a imparare. A scuola gli insegnanti sono invitati a utilizzare il coding, che è il modo più diffuso per favorire l'acquisizione del pensiero computazionale [8].

L'intelligenza artificiale (IA) svolge un ruolo sempre più importante nell'ambito della programmazione e delle discipline STEM; l'IA può fornire suggerimenti e assistenza agli studenti durante il processo di programmazione. Ad esempio, possono essere forniti suggerimenti sulla sintassi corretta, su come risolvere determinati problemi, o su possibili errori nel codice. Questo aiuta gli studenti a risolvere i problemi in modo più efficiente e a superare eventuali ostacoli. Gli studenti, inoltre, possono interagire con un tutor virtuale basato sull'IA che fornisce spiegazioni dettagliate, esempi di codice e risposte alle domande. Questo offre un supporto personalizzato e accessibile in qualsiasi momento, consentendo agli studenti di apprendere al proprio ritmo. L'IA può essere utilizzata per automatizzare attività ripetitive nella programmazione: ciò consente agli studenti di concentrarsi su compiti più creativi e complessi, migliorando la loro produttività. L'IA può essere utilizzata per generare automaticamente parti di codice o interi programmi. Gli studenti possono utilizzare questa funzionalità per ottenere un punto di partenza o per esplorare nuove soluzioni. L'IA può essere usata per insegnare agli studenti i principi dell'apprendimento automatico e dell'intelligenza artificiale stessa. Gli studenti possono sviluppare modelli di apprendimento automatico utilizzando librerie e framework specifici, esplorando algoritmi e concetti chiave [9 – 13].

2. Metodologia

Il percorso didattico proposto è strutturato in moduli e prevede sessioni laboratoriali svolte in un'aula multimediale o informatica. Durante queste sessioni, gli studenti acquisiscono gradualmente familiarità con i fondamenti del calcolo e della programmazione, apprendendo i comandi e la sintassi del linguaggio Python. Durante il periodo compreso tra l'a.s. 2021/2022 e il 2023, un gruppo di studenti del 4° anno dell'Istituto Tecnico Industriale Statale "Stanislao Cannizzaro" di Colleferro (RM) ha partecipato attivamente al progetto PON dal nome "Mettiamoci le Mani", promosso dal nostro istituto. Queste attività si sono svolte durante l'orario pomeridiano e sono state finanziate nel contesto del Programma Operativo Nazionale (PON E POC) "Per la scuola, competenze e ambienti per l'apprendimento" 2014-2021, con il codice del progetto: PROGETTO 10.2.2A-FSEPON-LA-2021-176, nell'ambito dell'Asse I - Istruzione - Obiettivi Specifici 10.1, 10.2 e 10.3 - Azioni 10.1.1, 10.2.2 e 10.3.1. Durante il progetto i ragazzi hanno sviluppato diversi codici utili, quali un convertitore di unità di misura, uno strumento per il calcolo del pH di acidi e basi deboli e forti e uno strumento per il calcolo della massa molecolare. L'obiettivo del progetto era anche quello di realizzare percorsi educativi volti al potenziamento delle competenze e all'aggregazione e socializzazione degli studenti nell'emergenza COVID-19.

L'uso di strumenti come Colab e Trinket si è rivelato fondamentale per superare le complessità tecniche legate all'ambiente di sviluppo tradizionale su un PC. Colab ha permesso agli studenti di scrivere il codice e vedere immediatamente i risultati, senza dover affrontare l'installazione del software Python o la gestione delle dipendenze. Inoltre, l'utilizzo di Colab ha semplificato la configurazione dell'ambiente di sviluppo e la gestione dei file di progetto.

Questo approccio ha favorito l'apprendimento pratico e interattivo, sviluppando le competenze chiave nel contesto delle discipline STEM. Gli studenti hanno potuto applicare le conoscenze teoriche in contesti pratici, promuovendo il pensiero logico-analitico, il problem solving e l'integrazione tra teoria e pratica nelle discipline scientifiche.

Colab, abbreviazione di Google Colaboratory, è un ambiente di sviluppo interattivo basato sul cloud che consente di scrivere, eseguire e condividere il codice Python. È un'opzione altamente conveniente per eseguire i codici Python precedentemente sviluppati, in quanto non richiede l'installazione di software sul proprio computer.

Per accedere a Colab, è sufficiente avere un account Google e visitare il sito web ufficiale di Google Colaboratory all'indirizzo colab.research.google.com. Una volta effettuato l'accesso, è possibile creare nuovi notebook o aprire notebook esistenti. I notebook sono file interattivi che combinano codice, testo descrittivo e risultati di esecuzione.

L'utilizzo di Colab è semplice ed intuitivo. È possibile scrivere il codice Python

all'interno delle celle di codice, eseguire ogni cella singolarmente o tutte le celle contemporaneamente, visualizzare i risultati di esecuzione, importare librerie e persino installare pacchetti aggiuntivi.

3. Il linguaggio Python - La sintassi

L'insegnamento della programmazione può essere una sfida, specialmente quando ci si trova di fronte a studenti disinteressati o poco competenti nella materia. Per affrontare questa sfida, è essenziale partire dalle funzioni basilari e dalla sintassi più semplice per introdurre Python.

Python è un linguaggio di programmazione versatile e potente, ma ciò non significa che debba essere insegnato in modo complesso fin dall'inizio. Al contrario, iniziare con le funzioni basilari e la sintassi più semplice può rendere l'apprendimento accessibile e coinvolgente per gli studenti che hanno poca familiarità con il coding e la programmazione.

Python è un linguaggio di programmazione ad alto livello, interpretato, dinamico e orientato agli oggetti. La sintassi di Python è progettata per essere chiara e intuitiva, facilitando così la scrittura e la comprensione del codice. La filosofia del linguaggio, spesso chiamata "Zen di Python", enfatizza la leggibilità del codice e favorisce l'approccio "esplicito è meglio del implicito". Ciò significa che il codice Python è solitamente scritto in modo chiaro e comprensibile, anche per i principianti.

Partire dalle funzioni basilari significa concentrarsi su concetti fondamentali come la dichiarazione delle variabili, la stampa dei messaggi, l'uso delle condizioni if-else e l'iterazione attraverso i cicli for. Questi concetti forniscono una base solida per la comprensione dei concetti più complessi che verranno introdotti successivamente.

La sintassi semplice, cioè l'organizzazione del codice in modo chiaro e comprensibile, è altrettanto importante. Spiegare agli studenti come strutturare correttamente il loro codice, utilizzando indentazioni adeguate e seguendo una logica di programmazione chiara, può aiutare a prevenire confusione e frustrazione.

Insegnare Python in modo graduale, partendo dalle basi e dalla sintassi più semplice, offre diversi vantaggi didattici. Prima di tutto, permette agli studenti di acquisire familiarità con il linguaggio gradualmente, senza sentirsi sopraffatti da concetti complessi. Inoltre, l'apprendimento progressivo consente agli studenti di sperimentare il successo fin dall'inizio, incoraggiandoli a continuare il loro percorso di apprendimento.

Se si vuole approfondire sono presenti diversi strumenti gratuiti on line per apprendere in modo significativo le basi del lingauggio Python come, ad esempio, i corsi Cisco "Skill for all" [14].

Oltre all'aspetto tecnico, è importante anche creare un ambiente di apprendimento stimolante e coinvolgente. Utilizzare esempi pratici, problemi reali e

progetti interessanti può aiutare a catturare l'attenzione degli studenti e dimostrare loro le applicazioni concrete della programmazione.

Infine, è fondamentale adattare l'insegnamento alle esigenze e agli interessi degli studenti. Cercare di comprendere le loro passioni e interessi personali e creare esempi e progetti che siano rilevanti per loro può rendere l'apprendimento di Python più significativo e coinvolgente.

In conclusione, partire dalle funzioni basilari e dalla sintassi più semplice per insegnare Python ai ragazzi disinteressati o poco competenti nella materia di coding e programmazione può essere un metodo efficace. Con un approccio graduale, un'organizzazione chiara del codice e l'utilizzo di esempi pratici e coinvolgenti, è possibile rendere l'apprendimento di Python accessibile, stimolante e gratificante per tutti gli studenti.

Ecco alcuni aspetti chiave della sintassi di Python che occorre introdurre per prendere pratica con i comandi.

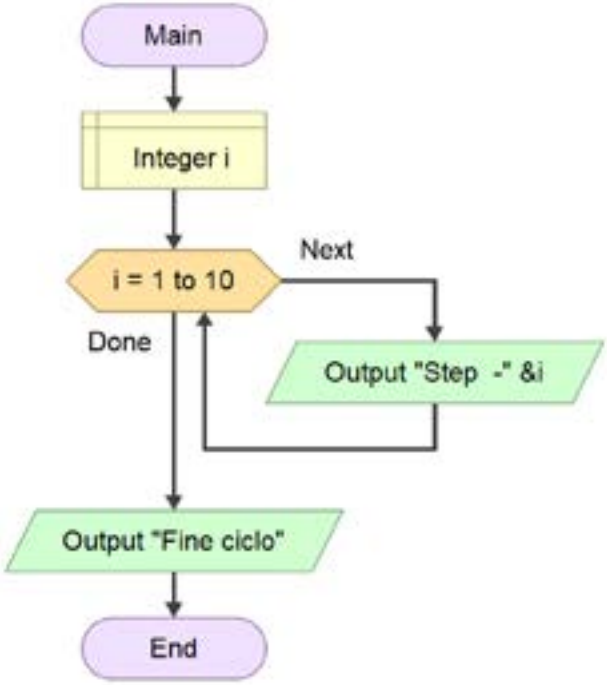
- *Indentazione:* Python utilizza l'indentazione per definire i blocchi di codice, come cicli, funzioni e classi. Gli spazi o le tabulazioni sono utilizzati per indicare la struttura del codice, evitando l'uso di parentesi graffe o altre delimitazioni simili.
- *Tipizzazione dinamica:* Python è un linguaggio a tipizzazione dinamica, il che significa che non è necessario dichiarare il tipo di una variabile esplicitamente. Le variabili in Python possono essere assegnate a diversi tipi di dati durante l'esecuzione del programma.
- *Commenti:* i commenti in Python vengono preceduti dal simbolo "#" e sono utilizzati per fornire spiegazioni o note nel codice. I commenti non vengono eseguiti e sono utili per rendere il codice più comprensibile agli altri programmatori.
- *Operatori:* di seguito è riportata una tabella con alcuni operatori matematici in Python

Operatore	Descrizione	Esempio	Risultato
+	Addizione	2 + 3	5
-	Sottrazione	5 – 3	2
*	Moltiplicazione	2 * 3	6
/	Divisione	6 / 3	2.0
//	Divisione intera	7 // 3	2
%	Resto della divisione	7 % 3	1
**	Potenza	2 ** 3	8
math.sqrt	Radice quadrata	math.sqrt(9)	3.0

Questi sono solo alcuni degli operatori matematici disponibili in Python. Puoi combinare questi operatori per eseguire calcoli più complessi.

- *Strutture dati*: Python fornisce diverse strutture dati incorporati, come liste, tuple, set e dizionari, che consentono di organizzare e manipolare i dati in modo efficiente.
- *Controllo del flusso*: Python offre costrutti per il controllo del flusso come if-else, cicli for e while, che consentono di eseguire azioni condizionali o ripetitive.

Un app utile è Flowgorithm [15] che ci consente di programmare attraverso schemi di flusso a blocchi e vedere il codice corrispondente ed eseguirlo direttamente:



Esempio di ciclo

```
for i in range(1, 10 + 1, 1):
print("Step  -" + str(i))
 print("Fine ciclo")
```

output:

```
Step  -1
Step  -2
Step  -3
Step  -4
Step  -5
Step  -6
Step  -7
Step  -8
Step  -9
Step  -10
Fine ciclo
```

- *Funzioni:* Le funzioni in Python sono definite utilizzando la parola chiave "def" e possono essere richiamate per eseguire azioni specifiche. Le funzioni consentono di modularizzare il codice e renderlo riutilizzabile.
  Esempio di funzione:

```
def saluta(nome):  # Definizione di una funzione 'saluta' con un parametro 'nome'
   print("Ciao,", nome, "! Benvenuto.")  # Stampa del messaggio di saluto
saluta("Marco")  # Chiamata alla funzione 'saluta' con il valore "Marco" come
argoment
```

Output:

```
Ciao, Marco! Benvenuto.
```

- *Librerie:* Python ha una vasta gamma di librerie e moduli che forniscono funzionalità aggiuntive. Queste librerie possono essere importate nel codice per estendere le capacità di base del linguaggio.
  Esempio di utilizzo di una libreria esterna:

```
import math  # Importazione della libreria 'math' per eseguire operazioni matematiche
raggio = 5  # Dichiarazione della variabile 'raggio' con il valore 5
area = math.pi * raggio ** 2  # Calcolo dell'area del cerchio utilizzando la costante pi
print("L'area del cerchio con raggio", raggio, "è", area)  # Stampa del risultato
```

Output:

```
L'area del cerchio con raggio 5 è 78.53981633974483.
```

4. Il codice sviluppato

Il percorso didattico sviluppato nella prima parte si basa sull'approccio di inquiry, che promuove l'apprendimento attraverso l'esplorazione attiva, la scoperta e l'indagine. In particolare, nella prima parte di questo percorso, gli studenti vengono guidati a risolvere problemi analitici in chimica utilizzando Python senza l'uso di moduli esterni.

Per avvicinare gli studenti al calcolo del pH di un acido forte e debole, vengono fornite loro le equazioni di base e i principi sottostanti all'equilibrio chimico. Inizialmente, gli esercizi si concentrano su sistemi abbastanza semplici e con approssimazioni che consentono agli studenti di risolvere gli equilibri analiticamente e ottenere una formula risolutiva per il calcolo del pH. Questo approccio classico è simile a quello che viene svolto tradizionalmente sulla lavagna e sul quaderno durante le lezioni di chimica. Tuttavia, come evidenziato, quando si affrontano sistemi chimici più complessi, come acidi poliprotici che coinvolgono equilibri multipli, la soluzione analitica diventa notevolmente più complessa. In queste situazioni, il calcolo del pH richiederebbe numerose approssimazioni e condizioni, rendendolo un esercizio di chimica estremamente ripetitivo e con poco valore didattico. A livello scolastico, quindi, è comune limitarsi a una descrizione qualitativa dei fenomeni più complessi come gli equilibri multipli.

Per superare questa limitazione e fornire agli studenti gli strumenti per affrontare anche problemi chimici più complessi, nella seconda parte di questo percorso didattico, ci si è concentrati sull'approccio di "thinkering" (pensare sperimentando) insieme all'utilizzo di moduli specializzati, come ChemPy, per

risolvere problemi chimici più complessi e realistici. Il thinkering è un concetto che combina il pensiero critico con l'esplorazione sperimentale e il tinkering (sperimentare e modificare). A differenza dell'approccio tradizionale, in cui gli studenti risolvono analiticamente gli equilibri chimici, il thinkering permette agli studenti di "armeggiare" con il codice Python e di sperimentare con diversi input e parametri per ottenere risultati interessanti. L'uso di moduli specializzati quale ChemPy si rivela prezioso quando si affrontano sistemi chimici complessi, come acidi poliprotici. Il principale vantaggio di questi moduli è la notevole semplificazione del processo di calcolo, il che consente agli studenti di concentrarsi sulla logica del problema e sulle regole per utilizzare correttamente tali strumenti. Questo approccio apre la possibilità di affrontare problemi chimici più realistici e complessi senza richiedere una conoscenza matematica estremamente approfondita, rendendo l'apprendimento più accessibile e stimolante. Anche se gli studenti potrebbero non comprendere pienamente tutti i dettagli matematici o gli algoritmi alla base dei moduli, acquisiranno comunque una competenza fondamentale nell'applicare strumenti tecnologici per esplorare e risolvere problemi scientifici. Questa metodologia offre un valore didattico significativo poiché consente agli studenti di affrontare in modo più efficace ed efficiente problemi complessi, sviluppando al contempo competenze di problem-solving e programmazione che saranno preziose sia nella scienza che in altre discipline.

Prima di sviluppare il codice per il calcolo del pH di un acido forte o debole, è stata effettuata una revisione della teoria chimica degli equilibri acido-base. Ci si è basati sulle formule e le approssimazioni presenti nel libro di testo in adozione alla classe "Chimica Analitica" [16]. Durante questo processo, sono state prese in considerazione le condizioni in cui alcune approssimazioni sono valide o non valide.

Ad esempio, nel caso di un acido forte, quando l'acidità è molto elevata e la concentrazione dell'acido è significativa, si può utilizzare direttamente il logaritmo della concentrazione per calcolare il pH.

Per un acido forte monoprotico HA si ha:

$$HA + H_2O \rightarrow H_3O^+ + A^-$$

Partendo dalla definizione di pH

$$pH = -\log_{10} a_{H_3O^+}$$

e considerando $\gamma \rightarrow 1$, si può calcolare il pH con la seguente espressione:

$$pH = -\log_{10} [H_3O^+]$$

Questa approssimazione è valida quando il contributo dell'autoionizzazione dell'acqua può essere trascurato rispetto alla concentrazione dell'acido forte e, cioè, quando la concentrazione dell'acido è superiore a 10^{-5} M.

D'altra parte, nel caso di una concentrazione più bassa dell'acido, si deve prendere in considerazione l'equilibrio di autoprotolisi dell'acqua con il suo valore K_w e considerare la relazione:

$$[H_3O^+]_{Totali} = [H_3O^+]_{acido} + [H_3O^+]_{acqua}$$

$2H_2O$	$\rightleftarrows$	H_3O^+	+	OH^-
		$[H_3O^+]_{acido} + [H_3O^+]_{acqua}$		$[OH^-]_{acqua}$
		$-x$		$-x$
		$[H_3O^+]_{acido} + [H_3O^+]_{acqua} - x$		$[OH^-]_{acqua} - x$

$$K_w = [H_3O^+] \times [OH^-]$$

$$K_w = ([H_3O^+]_{acido} + [H_3O^+]_{acqua} - x) \times ([OH^-]_{acqua} - x) = 10^{-14}$$

Risolvendo l'equazione di secondo grado possiamo ricavare e quindi calcolare il pH.

Per un acido debole monoprotico HA si ha:

$$HA + H_2O \rightleftarrows H_3O^+ + A^-$$

$$K_a = \frac{[H_3O^+] \times [A^-]}{[HA]}$$

C_a = *concentrazione analitica dell'acido*

$x = [H_3O^+] = [A^-]$

$$HA + H_2O \rightleftarrows H_3O^+ + A^-$$

C_a	–	–
$-x$	x	x
$C_a - x$	x	x

$$K_a = \frac{[H_3O^+]^2}{C_a - [H_3O^+]}$$

Se $C_a >> [H_3O^+] \quad \Rightarrow \quad C_a - [H_3O^+] \cong C_a$

$$K_a = \frac{[H_3O^+]^2}{C_a} \quad \Rightarrow \quad [H_3O^+] = \sqrt{K_a \, x \, C_a}$$

Altrimenti è necessario risolvere l'equazione di 2° grado:

$$[H_3O^+]^2 - K_a \times [H_3O^+] + K_a \times C_a = 0$$

Tuttavia, nel primo caso, è necessario tenere conto di alcune approssimazioni come, ad esempio, il fatto che la concentrazione dell'acido debole sia molto inferiore rispetto alla concentrazione dell'acqua e che il rapporto tra la concentrazione dell'acido e la costante di dissociazione sia inferiore a 100.

Considerando queste condizioni e approssimazioni, è stato possibile sviluppare un codice che permette di calcolare correttamente il pH di un acido forte o debole, fornendo così un utile strumento per gli studi di chimica analitica.

Durante il periodo di sviluppo del codice nel contesto del progetto "Mettiamoci le Mani", gli studenti hanno potuto beneficiare del supporto dell'Intelligenza Artificiale (IA) (OpenAI, s.d.) quale strumento di aiuto nella creazione degli algoritmi. L'IA ha fornito un supporto prezioso dando suggerimenti e soluzioni, ottimizzando il processo di sviluppo e consentendo agli studenti di affrontare in modo più efficace le sfide di programmazione. L'IA ha svolto un ruolo importante nel migliorare la produttività e la precisione degli algoritmi, consentendo agli studenti di concentrarsi sulle logiche di risoluzione del problema e di approfondire la comprensione dei concetti fisici e chimici sottostanti. Di seguito viene riportato il codice fornito nel riquadro a parte, con commenti che spiegano il funzionamento di ciascun codice.

4.1 Convertitore

Si parte da un semplice codice, un convertitore che prende in input un valore in metri cubi e lo converte in litri moltiplicando per 1000 (Figura 1). Il risultato viene quindi stampato a video.

```
#Convertitore.py
print('convertitore metri cubi a litri')
M_3 = float(input('inserisci il valore in metri cubi :'))
L = M_3 * 1000
print('il risultato è:', L, 'Litri')
```

*In appendice il commento al codice

Figura 1. Esempio di utilizzo del codice Convertitore in google Colab

4.2 Calcolo del pH di un acido forte

Questo codice calcola il pH di un acido forte in base alla sua molarità. L'utente può inserire la molarità dell'acido e il programma calcola il pH corrispondente (Figura 2). Se la molarità è inferiore a 10^{-6}, viene risolta un'equazione di secondo grado per ottenere il pH corretto. Se la molarità è maggiore o uguale a 10^{-6}, viene calcolato direttamente il pH utilizzando il logaritmo in base 10. L'utente può inserire 0 per uscire dal programma.

```
#Acidoforteloop.py
import math
print('calcolo pH di un acido Forte --- inserisci 0 per uscire')
pH = float(0)
Ca = 0.1
while Ca > 0:
   Ca = float(input('inserisci la Molarità dell acido: '))
   if Ca < 1e-6 and Ca > 0:
      print('Molto diluita!! risolvo l\'equazione di secondo grado')
      x = (-Ca + ((Ca**2) + 4e-14)**0.5) / 2
      x += Ca
      x = -math.log(x, 10)
      print('il pH è:', x)
   elif Ca >= 1e-6:
      pH = -math.log(Ca, 10)
      pH = round(pH, 2)
      print('il pH è:', pH)
else:
   print('Fine')
```

*In appendice il commento al codice

```
calcolo pH di un acido Forte --- inserisci 0 per uscire
inserisci la Molarità dell acido: 0.1
il pH è: 1.0
inserisci la Molarità dell acido: 0
Fine
```

Figura 2. Output del codice per il calcolo del pH di un acido forte

4.3 Calcolo del pH di un acido debole

Questo codice calcola il pH di un acido debole in base alla sua molarità e alla costante di acidità K_a. L'utente può inserire la molarità dell'acido e la K_a, e il programma calcola il pH corrispondente (Figura 3). Vengono considerati diversi casi: se la molarità divisa per la K_a è inferiore a 100 e la molarità è maggiore di 10^{-5}, viene risolta un'equazione di secondo grado per ottenere il pH corretto. Se la molarità divisa per la K_a è maggiore di 100, viene calcolato il pH utilizzando la radice quadrata del prodotto della molarità per la K_a. Se nessuna delle condizioni precedenti è soddisfatta, viene stampato un messaggio di avviso. L'utente può inserire 0 per uscire dal programma.

```
#acidodebole.py
import math
print('''----calcolo pH di un acido Debole ---
inserisci 0 per uscire ''')
pH = float(0)
Ca = 0.1
Ka = float(input('Inserisci la Ka: '))
while Ca > 0 and Ka > 0:
   Ca = float(input('inserisci la Molarità dell acido: '))
   if Ca == 0:
      break
   if Ca/Ka <= 100 and Ca >= 1e-5:
      print('Molto diluita o acido poco debole !! risolvo l\'equazione di secondo grado')
      x = (-Ka + ((Ka**2) + 4*Ka*Ca)**0.5) / 2
      x = -math.log(x, 10)
      print('il pH è:', x)
   elif Ca/Ka > 100:
      H = (Ca*Ka)**0.5
      pH = -math.log(H, 10)
      pH = round(pH, 2)
      print('il pH è:', pH)
   else:
      print('Troppo Diliuta....!')
      break
print('Fine')
```

*In appendice il commento al codice

```
----calcolo pH di un acido Debole ---
inserisci 0 per uscire
Inserisci la Ka: 10e-5
inserisci la Molarità dell acido: 0.01
Molto diluita o acido poco debole !! risolvo l'equazione di secondo grado
il pH è: 3.021705686457112
inserisci la Molarità dell acido: 0.1
il pH è: 2.5
inserisci la Molarità dell acido: 0
Fine
```

Figura 3. Esempio di utilizzo del codice per il calcolo del pH di un acido debole

5. Chempy e calcolo di equilibri multipli

L'utilizzo di librerie o strumenti specializzati come ChemPy [17] nel contesto dello sviluppo di codice per calcolare il pH di acidi può offrire diversi vantaggi significativi rispetto alla creazione manuale di algoritmi. Una volta compreso il meccanismo di base, l'adozione di ChemPy può semplificare notevolmente il processo di sviluppo e migliorare l'efficienza complessiva. Una sostanziale novità viene introdotta in questa fase nella programmazione con l'utilizzo delle librerie e della programmazione agli oggetti rispetto alla programmazione procedurale e l'utilizzo delle variabili finora utilizzati. La differenza fondamentale tra la programmazione procedurale utilizzando le variabili e la programmazione orientata agli oggetti sta nell'approccio e nell'organizzazione del codice. Nella programmazione procedurale, il codice è organizzato in procedure o funzioni che manipolano dati mediante l'utilizzo di variabili. Le variabili contengono i valori e vengono passate alle funzioni per elaborare i dati. Questo approccio è più lineare e mira a risolvere un problema attraverso una sequenza di passi strutturati, invece nella programmazione orientata agli oggetti, il codice è organizzato intorno agli "oggetti", che rappresentano entità del mondo reale con attributi e comportamenti specifici. Gli oggetti interagiscono tra loro attraverso messaggi e ogni oggetto può essere visto come una "capsula" che incorpora dati e funzioni pertinenti ad esso. Questo paradigma consente di organizzare il codice in modo più modulare, permettendo la riutilizzabilità del codice e una maggiore astrazione dei concetti.

Uno dei principali vantaggi di ChemPy è l'accesso ad algoritmi già pronti e specificamente progettati per gestire situazioni complesse con equilibri multipli. Questo è particolarmente importante quando si lavora con acidi deboli che possono avere molteplici specie ioniche e possono comportarsi in modo non lineare. ChemPy offre strumenti e funzioni specifiche per risolvere tali equilibri in modo accurato e preciso, fornendo risultati affidabili senza la necessità di implementare manualmente algoritmi complessi.

Inoltre, ChemPy permette l'utilizzo di database con dati sulle sostanze chimiche, come la massa molecolare, i dati termodinamici e altre proprietà rilevanti.

Questi database consentono di accedere rapidamente e facilmente a informazioni chiave sulle sostanze coinvolte negli equilibri acido-base. Ciò semplifica il processo di ricerca e recupero delle informazioni necessarie per effettuare i calcoli corretti e rende il codice più robusto e affidabile.

Per installare la libreria ChemPy è necessario eseguire in google colab il seguente comando prima di eseguire il codice. Una volta installata la libreria non sarà piu necessario eseguire di nuovo il comando

```
!pip install chempy.
```

Per i comandi fare riferimento alla guida on line di ChemPy al link: https://pythonhosted.org/chempy/

5.1 Calcolo del pH dell'ammoniaca

Questo codice calcola il pH di una soluzione di ammoniaca (NH_3) in equilibrio con ammonio (NH_4^+), in base alla concentrazione iniziale dei componenti (Figura 4). Viene utilizzata la libreria ChemPy per gestire l'equilibrio chimico. Viene definito un sistema di equazioni chimiche con le reazioni di autoprotolisi dell'acqua e di protolisi dell'ammoniaca. Viene quindi calcolato il pH utilizzando il metodo root del sistema di equazioni. Viene stampato il risultato a video.

```
#Ammonia.py
from collections import defaultdict
from chempy import Equilibrium
from chempy.equilibria import EqSystem
from chempy.chemistry import Species
from math import log10
Cb = float(input('Inserisci concentrazione molare ammoniaca: '))
water_autop = Equilibrium({'H2O'}, {'H+', 'OH-'}, 10**-14)  # unit "molar" assumed
ammonia_prot = Equilibrium({'NH4+'}, {'NH3', 'H+'}, 10**-9.24)  # same here
substances = [Species.from_formula(f) for f in 'H2O OH- H+ NH3 NH4+'.split()]
eqsys = EqSystem([water_autop, ammonia_prot], substances)
print('\n'.join(map(str, eqsys.rxns)))  # "rxns" short for "reactions"
init_conc = defaultdict(float, {'H2O': 1, 'NH3': Cb})
x, sol, sane = eqsys.root(init_conc)
assert sol['success'] and sane
print(' M, '.join('%.2g' % v for v in x))
print("pH: %.2f" % -log10(x[2]))
```

*In appendice il commento al codice

```
Inserisci concentrazione molare ammoniaca: 0.01
H2O = H+ + OH-; 1e-14
NH4+ = H+ + NH3; 5.75e-10
1 M, 0.00041 M, 2.4e-11 M, 0.0096 M, 0.00041
pH: 10.61
```

Figura 4. Esempio di utilizzo della libreria Chempy nel codice "ammonia.py"

5.2 Calcolo del pH di un acido monoprotico debole

Questo codice calcola il pH di un acido debole monoprotico (HA) che si dissocia in uno ione A^- e un protone H^+. L'utente può inserire i valori per HA, A^-, K_a (costante di acidità) per diversi tipi di acidi in una lista all'interno del codice o eventualmente in un modulo separato; successivamente viene richiesto di selezionare il tipo di acido e la sua concentrazione iniziale C_a (concentrazione iniziale di HA). Viene utilizzata la libreria chempy per gestire l'equilibrio chimico. Viene definito un sistema di equazioni chimiche con le reazioni di autoprotolisi dell'acqua e di protolisi dell'acido debole. Viene quindi calcolato il pH utilizzando il metodo root del sistema di equazioni (Figura 5). Viene stampato il risultato a video.

```
#monoprotico.py
from collections import defaultdict
from chempy import Equilibrium
from chempy.equilibria import EqSystem
from chempy.chemistry import Species

# Lista degli acidi monoprotici
acidi_monoprotici = {
   'Acido acetico': {'formula': 'CH3COOH', 'Ka': 1.8e-5},
   'Acido formico': {'formula': 'HCOOH', 'Ka': 1.8e-4},
   'Acido propionico': {'formula': 'C2H5COOH', 'Ka': 1.3e-5},
   'Acido butirrico': {'formula': 'C3H7COOH', 'Ka': 1.5e-5},
   'Acido valerico': {'formula': 'C4H9COOH', 'Ka': 1.8e-5},
   'Acido benzoico': {'formula': 'C6H5COOH', 'Ka': 6.3e-5}
   # Aggiungi altri acidi monoprotici alla lista con le relative formule e valori di Ka
}

print("Calcolo pH acido monoprotico")
print("Seleziona un acido monoprotico dalla lista:")
for i, acido in enumerate(acidi_monoprotici.keys(), start=1):
   print(f"{i}. {acido}")

scelta = int(input("Inserisci il numero corrispondente all'acido scelto: "))
acido_scelto = list(acidi_monoprotici.keys())[scelta - 1]
formula = acidi_monoprotici[acido_scelto]['formula']
```

```
Ka = acidi_monoprotici[acido_scelto]['Ka']
Ca = float(input("Inserisci la concentrazione di HA: "))

water_autop = Equilibrium({'H2O'}, {'H+', 'OH-'}, 10**-14)
acid_prot = Equilibrium({formula}, {f'{formula[:-1]}-', 'H+'}, Ka)

substances = [Species.from_formula(f) for f in ['H2O', 'OH-', 'H+', formula, f'{formu-
la[:-1]}-']]
eqsys = EqSystem([water_autop, acid_prot], substances)

init_conc = defaultdict(float, {'H2O': 55.4, formula: Ca})
x, sol, sane = eqsys.root(init_conc)

if sol['success'] and sane:
    conc = dict(zip(eqsys.substances, x))
    from math import log10
    pH = -log10(conc['H+'])
    print(f"pH: {pH:.2f}")
else:
    print("Impossibile calcolare il pH.")
```

*In appendice il commento al codice

```
Calcolo pH acido monoprotico
Seleziona un acido monoprotico dalla lista:
1. Acido acetico
2. Acido formico
3. Acido propionico
4. Acido butirrico
5. Acido valerico
6. Acido benzoico
Inserisci il numero corrispondente all'acido scelto: 2
Inserisci la concentrazione di HA: 0.01
pH: 2.90
```

Figura 5. Esempio di utilizzo della libreria Chempy nel codice "monoprotico.py"

6. pHcalc e curve di titolazione

pHcalc [18] è un modulo Python che fornisce funzionalità per il calcolo delle curve di titolazione acido-base. È uno strumento molto utile per gli scienziati e gli studenti di chimica che desiderano esplorare e comprendere il comportamento acido-base di diverse specie chimiche.

Con pHcalc è possibile definire acidi e basi con i loro pK_a, concentrazioni iniziali e cariche. Utilizzando un algoritmo basato sull'equilibrio chimico, pHcalc risolve le equazioni di bilancio ionico e calcola il pH delle soluzioni in diverse condizioni.

Questo modulo permette di tracciare le curve di titolazione acido-base, che mostrano come il pH di una soluzione varia al variare del volume di una sostanza titolante aggiunta. È possibile ottenere curve di titolazione per una vasta gamma di acidi e basi, consentendo di esplorare diversi scenari e confrontare i risultati. L'utilizzo di pHcalc semplifica il calcolo dei pH delle soluzioni durante le titolazioni e fornisce un modo pratico per visualizzare e analizzare i dati sperimentali. Inoltre, grazie alla flessibilità e alla modularità di Python, pHcalc può essere facilmente integrato in script personalizzati o applicazioni più ampie per l'analisi chimica.

pHcalc calcola il pH di un sistema complesso di acidi e basi utilizzando un metodo di soluzione dell'equilibrio sistemico. Questo metodo è descritto in dettaglio in un articolo del *Journal of Chemical Education* [19] e in uno di WikiChem [20]. Essenzialmente, questo metodo trova il pH ottimale per la miscela regolando sistematicamente il pH fino a raggiungere un equilibrio di carica, ovvero le concentrazioni degli ioni a carica positiva uguali a quelle degli ioni a carica negativa. Per gli acidi deboli, viene determinata la distribuzione frazionata delle specie a un dato valore di pH. Moltiplicando questa distribuzione per la concentrazione dell'acido in soluzione, si ottengono le concentrazioni di ciascuna specie acida nel sistema che vengono utilizzate per bilanciare la carica. Questa strategia risolutiva di solito non viene trattata con gli studenti delle scuole superiori di secondo grado, ma è necessario introdurla, partendo dalla condizione di elettroneutralità e dal bilancio di massa, per trattare i diagrammi di distribuzione. Per gli acidi deboli poliprotici viene determinata la distribuzione frazionata delle specie a un dato valore di pH. Questo processo coinvolge il bilancio di massa e il bilancio di carica delle specie presenti. Ad esempio, considerando un acido poliprotico come l'acido carbonico, H_2CO_3, che si dissocia in due ioni idrogeno, H^+, uno ione idrogenocarbonato, HCO_3^-, e uno ione carbonato, CO_3^{2-}, il pH ottimale è determinato tenendo conto delle concentrazioni di tutte le specie coinvolte e garantendo la conservazione della massa e della carica. Moltiplicando la distribuzione frazionata per la concentrazione dell'acido in soluzione, si ottiene la concentrazione di ciascuna specie acida nel sistema, e queste concentrazioni vengono utilizzate per bilanciare la carica. Usando tale metodologia, le basi e gli acidi forti possono essere descritti utilizzando specie inerti e cariche. Una combinazione 1:1 di Na^+ e H_2CO_3 descriverebbe una soluzione di $NaHCO_3$, e così via. Per installare la libreria pHcalc è necessario eseguire in google colab il segente comando prima di eseguire il codice; una volta installata la libreria non sarà piu necessario eseguire di nuovo il comando

```
!pip install pHcalc
```

Per i comandi fare riferimento all guida on line di pHcalc al link: https://pypi.org/project/pHcalc/

Oltre al modulo pHcalc è utilizzato il modulo Numpy [21]. Scipy e Numpy sono due librerie molto popolari e potenti in linguaggio di programmazione Python, utilizzate per elaborazioni numeriche, calcoli matematici e scientifici. Numpy offre potenti strutture dati e operazioni vettoriali, mentre Scipy fornisce funzionalità matematiche e scientifiche specializzate costruite sopra Numpy, consentendo agli utenti di Python di eseguire calcoli avanzati e complessi in modo più semplice ed efficiente.

Esempio di utilizzo di pHcalc.

pHcalc definisce tre classi - Acid, Inert e System - che vengono usate per calcolare il pH del sistema. H_3O^+ e OH^- non vengono definiti esplicitamente. La concentrazione di H_3O^+ viene regolata internamente e OH^- viene calcolato utilizzando K_w (Figura 6).

```
from pHcalc import Acid, Inert, System
import numpy as np
import matplotlib.pyplot as plt

# pH di 0.01 M di HCl
cl = Inert(charge=-1, conc=0.01)
system = System(cl)
system.pHsolve()
print(system.pH)

# pH di 1e-8 M di HCl
cl = Inert(charge=-1, conc=1e-8)
system = System(cl)
system.pHsolve()
print(system.pH)
```

*In appendice il commento al codice

```
1.9999977111816385
6.9782958984375005
```

Figura 6. Output dell'esempio

Utilizzando un semplice loop, è possibile costruire anche curve di titolazione arbitrarie (Figura 7). In questo esempio, si titola l'H_3PO_4 con NaOH.

```
import numpy as np
import matplotlib.pyplot as plt
from pHcalc import Acid, Inert, System

na_moles = np.linspace(1e-8, 5.e-3, 500)
sol_volume = 1. # Litri
phos = Acid(pKa=[2.148, 7.198, 12.375], charge=0, conc=1.e-3)
phs = []

for mol in na_moles:
    na = Inert(charge=1, conc=mol/sol_volume)
    system = System(phos, na)
    system.pHsolve(guess_est=True)
    phs.append(system.pH)

plt.plot(na_moles, phs)
plt.show()
```

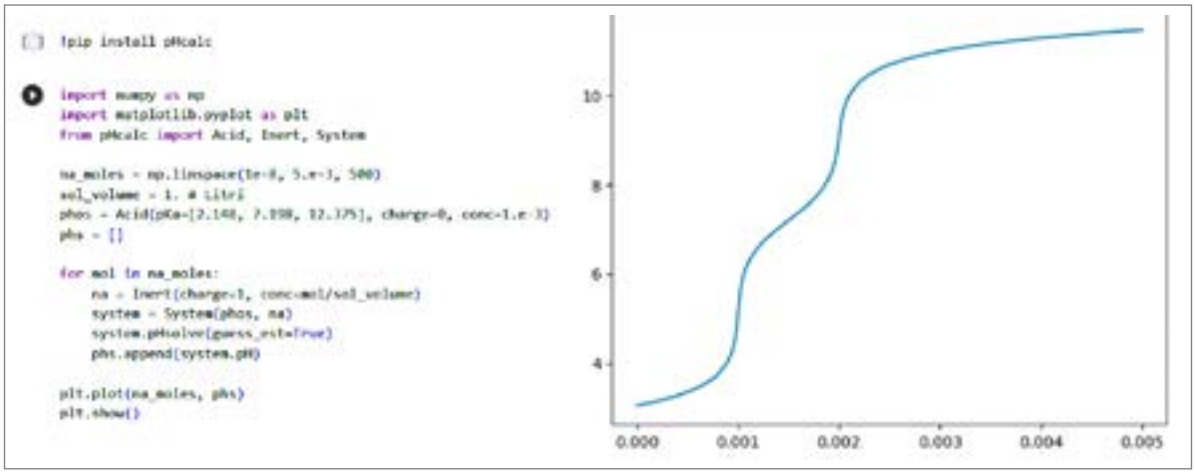


Figura 7. Esempio di costruzione di curve di titolazione con pHcalc

6.1 Modifica dell'algoritmo

Con l'aiuto di pHcalc e l'assistenza dell'IA di Chat GPT [22], siamo giunti a un punto in cui è diventato facile aggiungere funzionalità al nostro codice per tracciare curve di titolazione di diversi acidi. Come abbiamo visto precedentemente, pHcalc è una libreria Python che calcola il pH di un sistema complesso di acidi e basi, forti e deboli, utilizzando un metodo di soluzione degli equilibri sistematico. Grazie alla sua semplicità d'uso e alle sue potenti funzionalità, siamo stati in grado di estendere il codice per tracciare le curve di titolazione di acidi come l'acido fosforico, l'acido carbonico e molti altri. L'aggiunta delle curve di titolazione è stata possibile anche grazie all'interazione con l'IA di Chat GPT,

che ha fornito informazioni e spiegazioni sulle diverse classi e metodi di pHcalc. Inoltre, l'IA di Chat GPT ha facilitato la traduzione e la sintesi delle istruzioni di utilizzo e degli esempi di codice forniti nella documentazione di pHcalc.

Ora, con questa combinazione di strumenti, abbiamo la capacità di esplorare e visualizzare graficamente le curve di titolazione di diversi acidi, consentendoci di approfondire la comprensione dei processi chimici e delle relazioni tra concentrazioni e pH (Figura 8).

```
from pHcalc import Acid, Inert, System
import numpy as np
import matplotlib.pyplot as plt

def plot_titration_curve(acid, na_moles, sol_volume):
    phs = []
    for mol in na_moles:
        na = Inert(charge=1, conc=mol/sol_volume)
        system = System(acid, na)
        system.pHsolve(guess_est=True)
        phs.append(system.pH)
    plt.plot(na_moles, phs)
print("Disegna curva di titolazione dell'acido selezionato")
# Parametri comuni
na_moles = np.linspace(1e-8, 5.e-1, 500)
sol_volume = 1. # Liter

# Lista degli acidi
acids = [
    Acid(pKa=[2.148, 7.198, 12.375], charge=0, conc=1.e-1),  # Acido fosforico
    Acid(pKa=[1.21, 4,21], charge=0, conc=1.e-1),      # Acido ossalico
    Acid(pKa=[1.91, 6.07], charge=0, conc=1.e-1),      # Acido maleico
    Acid(pKa=[9.24, 12.4, 13.3], charge=0, conc=1.e-1),  # Acido borico
    Acid(pKa=[-3, 1.99], charge=0, conc=1.e-1),              # Acido solforico
    Acid(pKa=[4.76], charge=0, conc=1.e-1),              # Acido acetico
    Acid(pKa=[4.19], charge=0, conc=1.e-1),              # Acido benzoico
    Acid(pKa=[1.89,7.25], charge=0, conc=1.e-1),             # Acido solforoso
    Acid(pKa=[6.34,10.36], charge=0, conc=1.e-1),            # Acido carbonico
]

# Imposta i nomi degli acidi
acids[0].name = "Acido fosforico"
acids[1].name = "Acido ossalico"
acids[2].name = "Acido maleico"
acids[3].name = "Acido borico"
acids[4].name = "Acido solforico"
acids[5].name = "Acido acetico"
```

```
acids[6].name = “Acido benzoico”
acids[7].name = “Acido solforoso”
acids[8].name = “Acido carbonico”

# Scelta dell’acido
print(“Seleziona l’acido:”)
for i, acid in enumerate(acids):
    print(f”{i+1}. {acid.name}”)
choice = int(input(“Scelta: “))

if 1 <= choice <= len(acids):
    selected_acid = acids[choice - 1]
    plot_titration_curve(selected_acid, na_moles, sol_volume)
    plt.title(f”Curva di titolazione dell’{selected_acid.name} 0.1 M V=1L”)
    plt.xlabel(“Moli di NaOH”)
    plt.ylabel(“pH”)
    plt.show()
else:
    print(“Scelta non valida.”)
```

*In appendice il commento al codice

Il grafico della curva di titolazione verrà quindi visualizzato come di seguito riportato.

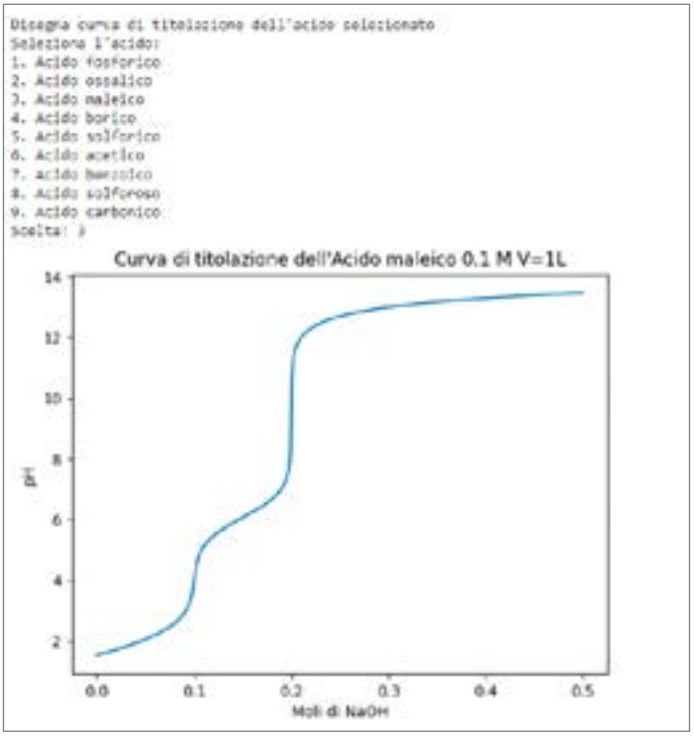


Figura 8. Curva titolazione dell’acido maleico

6.2 Sviluppi ulteriori

Un ulteriore sviluppo potenziale per il calcolo del pH di un acido monoprotico potrebbe essere l'utilizzo delle API di servizi esterni come Wikipedia [20] e PubChemPy [23] per ottenere informazioni aggiuntive sull'acido selezionato.

Le API (Application Programming Interface) sono un insieme di regole e protocolli che consentono a due software di comunicare tra loro. le API possono essere usate per ottenere informazioni aggiuntive da servizi esterni come Wikipedia, Google. L'utilizzo delle API di Wikipedia permette di recuperare descrizioni e informazioni chimiche sull'acido selezionato consentendo di arricchire l'applicazione con informazioni aggiuntive sull'acido e integrare dati provenienti da diverse fonti. PubChemPy è un servizio che fornisce informazioni sulla chimica, inclusi dati sulle proprietà delle sostanze chimiche. Potrebbe essere implementata una funzione che, utilizzando l'API di PubChemPy, cerca il composto chimico corrispondente all'acido selezionato e recupera informazioni come la sua struttura molecolare, peso molecolare, numero CAS e altre proprietà chimiche rilevanti. Oltre a Wikipedia e PubChemPy, potrebbero essere utilizzate altre API chimiche o scientifiche per ottenere ulteriori informazioni sull'acido, ad esempio API che forniscono dati sulla tossicità, reattività o effetti ambientali. Questi sviluppi possono essere implementati tramite richieste HTTP alle API pertinenti e l'elaborazione dei dati restituiti all'interno dell'applicazione.

Sviluppi futuri per le app basate sugli script di calcolo del pH potrebbero includere un'interfaccia utente intuitiva, un database di acidi completo, l'aggiunta di equilibri chimici più complessi, l'integrazione con servizi esterni per informazioni aggiuntive, personalizzazione delle unità di misura, salvataggio e condivisione dei risultati e il supporto per multiple soluzioni.

7. Conclusioni

In conclusione, il percorso che abbiamo seguito, partendo dal convertitore di unità di misura fino agli script per il calcolo del pH degli acidi monoprotici, ha offerto una panoramica completa delle potenzialità delle librerie chimiche e dei vantaggi di utilizzare un codice per semplificare e automatizzare i calcoli chimici. Questo tipo di approccio presenta numerosi risvolti didattici, consentendo agli studenti di chimica di sperimentare in prima persona come le nozioni teoriche possono essere applicate nella pratica. Attraverso l'utilizzo di librerie come ChemPy si è evidenziata l'importanza di un approccio modulare e basato sulle librerie per risolvere complessi problemi chimici. Questo approccio consente agli studenti di concentrarsi sulla comprensione dei concetti chimici fondamentali, mentre l'implementazione pratica viene semplificata grazie alle funzionalità offerte dalle librerie. Inoltre, l'integrazione di API esterne come Wikipedia e PubChem può aprire ulteriori opportunità per l'apprendimento. Gli studenti possono accedere a informazioni chimiche aggiun-

tive, esplorare proprietà molecolari, studiare reazioni chimiche e ampliare la loro comprensione del mondo chimico in generale. Il feedback riscontrato dal docente in aula dagli studenti è risultato inizialmente diffidente verso la nuova materia (coding e informatica), che non faceva parte del loro corso di studi in chimica e materiali, ed è stata percepita come un elemento estraneo. Tuttavia, con il passare del tempo, la programmazione è diventata sempre più interessante per loro, suscitando crescente curiosità e coinvolgimento. Verso la fine del modulo, gli studenti hanno cambiato la percezione del coding, non considerandolo più un elemento estraneo, ma comprendendone le potenzialità. Come previsto dalla piattaforma PON INdire, sono stati redatti dai tutor dei questionari iniziali e finali per rilevare l'atteggiamento dei partecipanti nei confronti del percorso di studi, nonché un questionario di gradimento specifico per il modulo seguito "Mettiamoci le mani". Dall'analisi dei dati emerge chiaramente un miglioramento dell'atteggiamento nei confronti dello studio e un discreto gradimento da parte dei partecipanti riguardo alla loro partecipazione al modulo. Questo tipo di approccio favorisce l'apprendimento attivo in cui gli studenti possono esplorare, scoprire e approfondire le proprie conoscenze. Infine, il potenziale sviluppo di app basate su questi script offre un'opportunità per creare strumenti interattivi che rendono l'apprendimento chimico ancora più coinvolgente e accessibile: gli studenti possono interagire direttamente con gli script, personalizzare i parametri, visualizzare grafici e risultati in modo intuitivo e sperimentare con diverse configurazioni chimiche. Quindi, tale approccio favorisce l'autonomia, la creatività e la comprensione profonda dei concetti chimici. In definitiva, l'utilizzo di script per il calcolo del pH e l'integrazione di librerie chimiche e API esterne offrono un'opportunità unica per rendere l'apprendimento della chimica più stimolante, pratico e coinvolgente.

Riferimenti

[1] Piano Nazionale Scuola Digitale (PNSD); available at: https://www.istruzione.it/scuola_digitale/index.shtml.

[2] G. Van Rossum, *Python library reference*, CWI report, 1995, https://ir.cwi.nl/pub/5009/05009D.pdf.

[3] G. Van Rossum, Python Programming language, in *USENIX annual technical conference*, 2007.

[4] B. Dahlgren, ChemPy: A package useful for chemistry written in Python, *Journal of Open Source Software,* 2018, 3(24), 565.

[5] F. Nati, Python ed il suo utilizzo nella ricerca scientifica, in *Linux Day*, Roma, 2002.

[6] P. Barabotti, L. Marcolini, Il computer.....bilancia le equazioni chimiche!, *CnS, La Chimica nella scuola,* 2002, 3, 88-90.

[7] E. J. Menke, Series of Jupyter Notebooks Using Python for an Analytical Chemistry Course, *Journal of Chemical Education,* 2007, 97(10), 3899-3903.

[8] L. Lana, V. Mazzoli, Il coding e le sue potenzialità didattiche, *Educare.it,* 2021, 21(9), 98-105.

[9] B. P. Woolf, *Building intelligent interactive tutors: student-centered strat egies for revolutionizing e-learning*, Morgan Kaufmann-Elsevier, 2008.

[10] A. Mitrovic, A. Weerasinghe, Towards Individualized Dialogue Support for Ill-Defined Domains, *International Journal of Artificial Intelligence in Education,* 2009, 19, 357-359.

[11] P. Slovak, G. Fitzpatrick, Teaching and developing social and emotional skills with technology, *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2015, 22(4), 1-34.

[12] M. Fahimirad, S. S. Kotamjani, A review on application of artificial intelligence in teaching and learning in educational contexts, *International Journal of Learning and Development*, 2018*,* 8(4), 106-118.

[13] P. Rospigliosi, Artificial intelligence in teaching and learning: what questions should we ask of ChatGPT?, *Interactive Learning Environments,* 2023, 31(1), 1-3.

[14] The Cisco Networking Academy platform, *Python essentials 1*; available at: https://skillsforall.com/course/python-essentials-1?courseLang=en-US

[15] *Flowgorithm*; available at: http://www.flowgorithm.org/

[16] L. F. Adelaide Crea, *Chimica Analitica-Analisi Qualititativa e quantitativa*, Zanichelli, 1999.

[17] *Chempy*; available at: https://pythonhosted.org/chempy/

[18] *pHcalc*; available at: https://pypi.org/project/pHcalc/

[19] J. J. Baeza-Baeza, M. C. García-Álvarez-Coque, Systematic Approach to Calculate the Concentration of Chemical Species in Multi-Equilibrium Problems, *Journal of Chemical Education,* 2011, 88(2), 169-173.

[20] *ChemWiki*; available at: http://chemwiki.ucdavis.edu/Core/Analytical_Chemistry/Analytical_Chemistry_2.0/06_Equilibrium_Chemistry/6G%3A_Solving_Equilibrium_Problems#6G.3_A_Systematic_Approach_to_Solving_Equilibrium_Problems

[21] E. Bressert, SciPy and NumPy: an overview for developers; available at: http://geofaculty.uwyo.edu/neil/teaching/Numerical_web/SciPy-NumPy.pdf

[22] OpenAI, CHATGPT; available at: https://chat.openai.com/

[23] Pubchem; available at: https://pubchempy.readthedocs.io/en/latest/

APPENDICE

In questa appendice sono raccolte le informazioni aggiuntive sui codici utilizzati.

Convertitore

Il codice inizia con l'istruzione `print('convertitore metri cubi a litri')` che stampa a video un messaggio per informare l'utente sulla funzione del convertitore.
Successivamente, viene utilizzata l'istruzione `M_3 = float(input('inserisci il valore in metri cubi :'))` per richiedere all'utente di inserire il valore in metri cubi da convertire. L'input viene letto come una stringa e viene convertito in un numero in virgola mobile utilizzando la funzione `float()`.
A seguire, l'istruzione `L = M_3 * 1000` esegue la conversione del valore in metri cubi in litri moltiplicando il valore per 1000, poiché ci sono 1000 litri in un metro cubo.
Infine, l'istruzione `print('il risultato è:', L, 'Litri')` stampa a video il risultato della conversione, concatenando il valore di `L` con del testo descrittivo.
In breve, il codice richiede all'utente di inserire un valore in metri cubi, lo converte in litri moltiplicando per 1000 e restituisce il risultato stampato a video.

Calcolo pH di un acido forte

Ecco una spiegazione linea per linea del codice
`import math`: questa linea importa il modulo `math`, che fornisce funzioni matematiche avanzate, come il logaritmo.
`print('calcolo pH di un acido Forte --- inserisci 0 per uscire')`: stampa un messaggio di testo che informa l'utente sull'obiettivo del programma e fornisce istruzioni su come uscire.
`pH = float(0)`: inizializza la variabile `pH` con il valore 0.0. Sarà utilizzata per memorizzare il pH calcolato.
`Ca = 0.1`: inizializza la variabile `Ca` con il valore 0.1. Sarà utilizzata per memorizzare la molarità dell'acido inserita dall'utente.
`while Ca > 0`: inizia un ciclo while che si ripeterà finché il valore di `Ca` è maggiore di zero. Questo permette all'utente di inserire più valori di molarità.
`Ca = float(input('inserisci la Molarità dell acido: '))`: richiede all'utente di inserire la molarità dell'acido e la converte in un numero in virgola mobile utilizzando la funzione `float()`. Il valore inserito viene assegnato alla variabile `Ca`.
`if Ca < 1e-6 and Ca > 0`: verifica se la molarità `Ca` è inferiore a 1e-6 (0.000001) e maggiore di zero. Questa condizione controlla se l'acido è molto diluito e richiede la risoluzione di un'equazione di secondo grado.
`print('Molto diluita!! risolvo l\'equazione di secondo grado')`: stampa un messaggio che indica che l'acido è molto diluito e che sarà risolta un'equazione di secondo grado per calcolare il pH corretto.

x = (-Ca + ((Ca2) + 4e-14)**0.5) / 2**: calcola il valore di **x** utilizzando l'equazione di secondo grado. Questa formula è una soluzione approssimata dell'equazione e permette di calcolare il pH corretto per acidi molto diluiti.
x += Ca: aggiunge il valore di **Ca** a **x** per ottenere il pH corretto.
x = -math.log(x, 10): calcola il logaritmo in base 10 di **x** utilizzando la funzione **math.log()** e lo assegna di nuovo a **x**. Questo calcola il pH effettivo.
print('il pH è:', x): stampa il valore del pH calcolato.
elif Ca >= 1e-6: se la molarità **Ca** è maggiore o uguale a 1e-6, esegue questo blocco di codice.
pH = -math.log(Ca, 10): calcola direttamente il pH utilizzando il logaritmo in base 10 della molarità **Ca** e lo assegna alla variabile **pH**.
pH = round(pH, 2): arrotonda il valore del pH a due cifre decimali utilizzando la funzione **round()** e lo assegna di nuovo a **pH**.
print('il pH è:', pH): stampa il valore del pH calcolato.
else: se la condizione del ciclo while (Ca > 0) non è soddisfatta, esegue questo blocco di codice.
print('Fine'): stampa un messaggio per indicare che il programma è terminato.

Calcolo pH ammoniaca

Ecco il commento linea per linea del codice
from collections import defaultdict
from chempy import Equilibrium
from chempy.equilibria import EqSystem
from chempy.chemistry import Species
Queste istruzioni importano le librerie necessarie per il codice. **defaultdict** viene utilizzato per creare un dizionario con valori predefiniti. **Equilibrium** è una classe per rappresentare un equilibrio chimico. **EqSystem** è una classe che rappresenta un sistema di equilibri. **Species** rappresenta una specie chimica.
water_autop = Equilibrium({'H2O'}, {'H+', 'OH-'}, 10-14) ammonia_prot = Equilibrium({'NH4+'}, {'NH3', 'H+'}, 10**-9.24)**
Qui vengono definiti due equilibri chimici: uno tra acqua, ioni H^+ e ioni OH^- e l'altro tra ammonio (NH_4^+), ammoniaca (NH_3) e ioni H^+. I valori di 10^{-14} e $10^{-9.24}$ rappresentano le costanti di equilibrio riferite alla dissociazione dell'acqua e all'idrolisi dell'ione ammonio. Entrambi i comandi seguono una struttura simile: si specificano le specie chimiche coinvolte nell'equilibrio come insiemi di stringhe, utilizzando le formule chimiche corrispondenti. Successivamente, si specificano i prodotti dell'equilibrio come un secondo insieme di stringhe. Infine, si specifica la costante di equilibrio, che rappresenta il rapporto tra le concentrazioni dei prodotti e dei reagenti all'equilibrio. Questi comandi sono utili per creare oggetti di equilibrio chimico che possono essere utilizzati per calcolare le concentrazioni finali delle specie chimiche coinvolte in un sistema di equilibrio.

substances = [Species.from_formula(f) for f in 'H2O OH- H+ NH3 NH4+'.split()]
Viene creato un elenco di oggetti **Species** corrispondenti alle specie chimiche coinvolte negli equilibri. Viene utilizzata la funzione **from_formula** per creare gli oggetti **Species** a partire dalle formule chimiche delle specie coinvolte.
A questo punto, definite le costanti di dissociazione e le variabili coinvolte, deve essere definito il sistema di equazioni
eqsys = EqSystem([water_autop, ammonia_prot], substances)
Qui viene creato un oggetto **EqSystem** che rappresenta il sistema di equilibri. Vengono passati gli equilibri chimici e le specie chimiche coinvolte.
print('\n'.join(map(str, eqsys.rxns)))
Questa istruzione stampa a video le reazioni chimiche presenti nel sistema di equilibri.
init_conc = defaultdict(float, {'H2O': 1, 'NH3': 0.1})
Viene creato un dizionario **init_conc** con le concentrazioni iniziali delle specie chimiche. La funzione **defaultdict** viene utilizzata per impostare il valore predefinito a zero.
x, sol, sane = eqsys.root(init_conc) assert sol['success'] and sane
Qui viene chiamato il metodo root dell'oggetto **EqSystem** per calcolare il punto di equilibrio che rappresenta un metodo computazionale implementato in Chempy per trovare le radici del sistema. Il risultato viene assegnato alle variabili **x, sol** e **sane**. Viene utilizzata un'asserzione per verificare che il calcolo abbia avuto successo e che il risultato sia valido.
print(' M, '.join('%.2g' % v for v in x))
print("pH: %.2f" % -log10(x[2]))
Infine, vengono stampati a video il risultato del calcolo delle concentrazioni al punto di equilibrio e il pH con i valori approssimati a due cifre significative.

Calcolo pH acido monoprotico debole

Nel codice per il calcolo del pH di un acido monoprotico debole, rispetto a **ammonia.py**, sono utilizzati i seguenti comandi aggiuntivi o modificati.
Utilizzo del modulo **chempy.equilibria** per gestire gli equilibri chimici tra le specie chimiche coinvolte nell'acido monoprotico.
Creazione di un dizionario chiamato **acidi_monoprotici** per memorizzare le informazioni sugli acidi monoprotici disponibili, come i loro nomi, formule e valori di K_a.
Presentazione all'utente di un elenco di acidi monoprotici disponibili e richiesta di selezionarne uno mediante l'inserimento del numero corrispondente.
Utilizzo del modulo **defaultdict** dalla libreria **collections** per gestire le concentrazioni iniziali delle specie chimiche coinvolte nell'equilibrio chimico.
Creazione dell'equilibrio chimico dell'acido monoprotico selezionato utilizzando la formula dell'acido e la sua forma ionica negativa (ottenuta rimuovendo l'atomo di idrogeno).

Gestione dell'eccezione **ValueError** nel caso in cui il calcolo del pH non sia possibile a causa di un acido non valido o di problemi durante il calcolo dell'equilibrio chimico.
Questi comandi aggiuntivi o modificati consentono di gestire in modo dinamico diversi acidi monoprotici e di calcolarne il pH in base alla selezione dell'utente.

pHcalc e curve di titolazione

Per i comandi fare riferimento alla guida on line di pHcalc al link: https://pypi.org/project/pHcalc/
Codice di esempio
Il codice fornito utilizza la libreria pHcalc per calcolare il pH di due diverse soluzioni di acido cloridrico (HCl) a diverse concentrazioni. Di seguito, viene fornito un commento linea per linea per spiegare il funzionamento del codice.

```
# Import delle librerie necessarie
from pHcalc import Acid, Inert, System
import numpy as np
import matplotlib.pyplot as plt
# Definizione di una soluzione di HCl con una concentrazione di 0.01 M
# In questo caso, HCl si dissocia completamente in acqua producendo quantità uguali di H3O+ e Cl-.
# Poiché H3O+ viene regolato internamente, è sufficiente definire solo Cl-.
# Ciò implica un equivalente di H3O+ per bilanciare la carica del sistema.
cl = Inert(charge=-1, conc=0.01)
system = System(cl)
# Risoluzione del sistema per determinare il pH
system.pHsolve()
# Stampa del valore del pH calcolato
print(system.pH) # Dovrebbe stampare 1.9999
# Definizione di una soluzione di HCl con una concentrazione di 1e-8 M
# Questo è un esempio notoriamente complicato per gli studenti di chimica introduttiva, ma pHcalc lo gestisce in modo eccellente.
cl = Inert(charge=-1, conc=1e-8)
system = System(cl)
# Risoluzione del sistema per determinare il pH
system.pHsolve()
# Stampa del valore del pH calcolato
print(system.pH) # il pH calcolato è 6.978295898 e (NON 8!)
```

Nell'esempio di utilizzo vengono creati oggetti delle classi **Acid**. **Inert** e **System**. La concentrazione dell'acido cloridrico (HCl) viene fornita come input per il

calcolo del pH. L'oggetto **Inert** rappresenta lo ione cloruro (Cl^-) nel sistema, mentre l'oggetto **System** viene istanziato con l'oggetto **Inert**.
Successivamente, utilizza il metodo **pHsolve()** per calcolare il pH del sistema, che tiene conto delle reazioni di dissociazione dell'acido cloridrico e del bilancio di carica. Il risultato del pH calcolato viene quindi stampato a schermo.
L'esempio dimostra come **pHcalc** semplifichi notevolmente il calcolo del pH, anche in situazioni complesse come la concentrazione molto bassa di HCl. Questo strumento può essere prezioso per gli studenti di chimica introduttiva che vogliono affrontare problemi complessi in modo più efficiente ed efficace.
Diversi codici di esempio sono reperibili alla pagina https://pypi.org/project/pHcalc/

pH di acidi poliprotici

Questo codice utilizza la libreria pHcalc per calcolare il pH di una soluzione contenente un acido poliprotico (con tre pKa) e un catione sodio (Na+) a diverse concentrazioni. Di seguito, viene fornito un commento linea per linea per spiegare il funzionamento del codice.

```
# Import delle librerie necessarie
import numpy as np
import matplotlib.pyplot as plt
from pHcalc import Acid, Inert, System

# Creazione di un array di 500 valori equidistanti nell'intervallo da 1e-8 a 5e-3 moles di Na+
na_moles = np.linspace(1e-8, 5.e-3, 500)

# Volume della soluzione in litri (1 litro)
sol_volume = 1.

# Definizione dell'acido poliprotico con i pKa specificati e la concentrazione di 1e-3 M
phos = Acid(pKa=[2.148, 7.198, 12.375], charge=0, conc=1.e-3)

# Creazione di una lista vuota per memorizzare i valori del pH calcolati
phs = []

# Ciclo for per iterare attraverso le diverse concentrazioni di Na+
for mol in na_moles:
    # Creazione di un oggetto Inert per il catione sodio Na+

    # La concentrazione viene calcolata dividendo il numero di moli (mol) per il volume della soluzione
    na = Inert(charge=1, conc=mol/sol_volume)
```

```
    # Creazione di un sistema contenente l'acido poliprotico (phos) e il catione sodio (na)
    system = System(phos, na)

    # Risoluzione del sistema per determinare il pH, utilizzando l'opzione 'guess_est' per stimare il pH iniziale
    system.pHsolve(guess_est=True)

    # Aggiunta del valore del pH calcolato alla lista 'phs'
    phs.append(system.pH)

# Creazione di un grafico che mostra come il pH varia al variare della concentrazione di Na+
plt.plot(na_moles, phs) # Vettori dei dati del grafico
plt.xlabel('Concentrazione Na+ (mol/L)') #etichetta asse X
plt.ylabel('pH') #etichetta asse Y
plt.title('Variazione del pH con la concentrazione di Na+')#titolo del grafico
plt.show() # comando che visualizza il grafico
```

Nell'esempio sopra, viene costruita una curva di titolazione dell'H_3PO_4 con NaOH. Viene utilizzato un loop per variare la concentrazione del NaOH (rappresentato dall'oggetto **Inert**) Per ogni valore di concentrazione di Na^+, viene creato un sistema chimico con l'acido poliprotico e il catione sodio, quindi viene risolto il sistema per calcolare il pH utilizzando l'opzione **'guess_est'** con la funzione **pHsolve()**. I valori del pH vengono salvati nella lista "**phs**". Infine, i valori del NaOH e del pH vengono tracciati utilizzando la funzione **plt.plot()** di matplotlib. La curva di titolazione viene visualizzata utilizzando **plt.show()**.

Curva titolazione acido

```
from pHcalc import Acid, Inert, System
import numpy as np
import matplotlib.pyplot as plt

def plot_titration_curve(acid, na_moles, sol_volume):
    phs = []
    for mol in na_moles:
        na = Inert(charge=1, conc=mol/sol_volume)
        system = System(acid, na)
        system.pHsolve(guess_est=True)
        phs.append(system.pH)
    plt.plot(na_moles, phs)
print("Disegna curva di titolazione dell'acido selezionato")
# Parametri comuni
na_moles = np.linspace(1e-8, 5.e-1, 500)
sol_volume = 1. # Liter
```

```
# Lista degli acidi
acids = [
    Acid(pKa=[2.148, 7.198, 12.375], charge=0, conc=1.e-1),  # Acido fosforico
    Acid(pKa=[1.21, 4,21], charge=0, conc=1.e-1),      # Acido ossalico
    Acid(pKa=[1.91, 6.07], charge=0, conc=1.e-1),      # Acido maleico
    Acid(pKa=[9.24, 12.4, 13.3], charge=0, conc=1.e-1),  # Acido borico
    Acid(pKa=[-3, 1.99], charge=0, conc=1.e-1),              # Acido solforico
    Acid(pKa=[4.76], charge=0, conc=1.e-1),              # Acido acetico
    Acid(pKa=[4.19], charge=0, conc=1.e-1),              # Acido benzoico
    Acid(pKa=[1.89,7.25], charge=0, conc=1.e-1),             # Acido solforoso
    Acid(pKa=[6.34,10.36], charge=0, conc=1.e-1),            # Acido carbonico
]

# Imposta i nomi degli acidi
acids[0].name = "Acido fosforico"
acids[1].name = "Acido ossalico"
acids[2].name = "Acido maleico"
acids[3].name = "Acido borico"
acids[4].name = "Acido solforico"
acids[5].name = "Acido acetico"
acids[6].name = "Acido benzoico"
acids[7].name = "Acido solforoso"
acids[8].name = "Acido carbonico"

# Scelta dell'acido
print("Seleziona l'acido:")
for i, acid in enumerate(acids):
    print(f"{i+1}. {acid.name}")
choice = int(input("Scelta: "))

if 1 <= choice <= len(acids):
    selected_acid = acids[choice - 1]
    plot_titration_curve(selected_acid, na_moles, sol_volume)
    plt.title(f"Curva di titolazione dell'{selected_acid.name} 0.1 M V=1L")
    plt.xlabel("Moli di NaOH")
    plt.ylabel("pH")
    plt.show()
else:
    print("Scelta non valida.")
```

In questo codice:

- viene importato il modulo necessario pHcalc, insieme alle librerie **numpy** e **matplotlib** per il calcolo e la visualizzazione dei dati;
- viene definita la funzione "**plot_titration_curve**" che prende come argomenti l'oggetto "acid" (un'istanza della classe Acid), "**na_moles**" (una sequenza di valori che rappresentano le moli di NaOH) e "**sol_volume**" (il volume della soluzione);
- all'interno della funzione "**plot_titration_curve**", viene eseguito un ciclo su "**na_moles**" per creare diverse combinazioni di acido e NaOH;
- viene istanziato un oggetto Inert per NaOH e un oggetto System che contiene l'acido e NaOH;
- viene quindi chiamato il metodo "**pHsolve**" dell'oggetto System per calcolare il pH della soluzione;
- infine, il valore del pH viene aggiunto alla lista "**phs**";
- viene definita una lista di acidi utilizzando l'istanza della classe Acid per ciascun acido desiderato; ogni acido è configurato con i suoi pK_a, carica e concentrazione iniziale;
- vengono impostati i nomi degli acidi nella lista "**acids**" utilizzando l'attributo "**name**" dell'oggetto Acid corrispondente;
- viene visualizzato un elenco degli acidi disponibili e viene richiesto all'utente di selezionarne uno;
- se la scelta è valida, viene eseguita la funzione "**plot_titration_curve**" passando l'acido selezionato, la sequenza di moli di NaOH e il volume della soluzione;
- viene quindi disegnata la curva di titolazione corrispondente utilizzando il modulo **matplotlib**;
- se la scelta non è valida, viene visualizzato un messaggio di errore.