**COME FUNZIONA IL CODICE**

Il codice presenta vari commenti (introdotti da # ed evidenziati in verde scuro) che spiegano le linee di codice in cui sono inseriti. Il commento continua per tutta la linea ed è una parte di codice che non viene eseguita. Nonostante i commenti, può risultare difficoltoso leggere e capire interamente il codice se non ci si è mai approcciati a Python o ad altri linguaggi di programmazione: per tale motivo segue una schematica guida, con integrazione di immagini, per interpretare il funzionamento del codice della calcolatrice presentata, oltre che per rispettare le linee guida di Unica sul caso in cui vengano usati collegamenti esterni.

Tra i collegamenti esterni è presente un file .exe singolo per scaricare ed eseguire direttamente il programma della calcolatrice.

**Creare la finestra**

Nella prima linea del codice, vengono importati i moduli e le funzioni da una libreria, per avere più funzioni con cui lavorare. La libreria in questione è “Tkinter”, utile per la creazione di interfacce grafiche (GUI) in Python.

Avendo importato il modulo, bisogna lavorare su come mostrare pulsanti e testo sullo schermo. Alla linea 190 viene creato l’oggetto principale *finestra* che conterrà le informazioni da mostrare. Alle linee successive (da linea 191 a 195) vengono attribuite delle caratteristiche all’oggetto *finestra*: dimensioni, colore dello sfondo, titolo, icona.

L’ultima linea (linea 296) serve a creare un loop che tenga effettivamente aperta la finestra fino al momento in cui l’utente non chiuda il programma.

**Alcune variabili fondamentali**

Una variabile è un identificatore che fa riferimento a un valore memorizzato. Tra la linea 197 e la 200, vengono dati i valori iniziali di alcune variabili che saranno fondamentali per il corretto funzionamento della calcolatrice. All’inizio le variabili sono vuote poiché vengono poste uguali a “”.

* *equazione\_testo:* rappresenta l’intera equazione ed è ciò che verrà poi calcolato nel risultato.
* *risultato*: come dice il nome, contiene il risultato finale dell’equazione.
* *testo*: contiene ciò che appare nella casella di testo in alto. Contiene l’input numerico dato dai pulsanti.
* *testo\_piccolo*: contiene l’equazione generale da aggiornare con il contenuto di *testo* ed è posto sopra di esso.

**Mostrare il contenuto delle variabili**

Le variabili servono solo a contenere ciò che verrà mostrato: da sé non mostrano il loro contenuto sullo schermo. Per farlo (da linea 202 a 208) bisogna introdurre la classe *Label()*: rende possibile visualizzare un’immagine o un testo in *finestra* con cui l’utente non interagisce direttamente. In questa classe vengono definite le caratteristiche del testo che verrà mostrato: font, dimensioni, colore, allineamento.

Il testo mostrato nell’etichetta è definito da *StringVar()*, una classe di Tkinter che rappresenta una variabile che può essere utilizzata per aggiornare in modo dinamico il contenuto dell’etichetta. Avendo due tipi di testi da mostrare, di cui il contenuto si trova nelle variabili sopraindicate, vengono definite due etichette e due variabili di testo correlate:

* La prima etichetta *etic* rimanda alla variabile di testo *testo\_eticchetta*. È quella più in alto con il testo più piccolo.
* La seconda etichetta *etichetta* rimanda alla variabile di testo *equazione\_etichetta*. È posta subito sotto a *etic*.

Le etichette vengono posizionate nella finestra tramite l’apposita funzione *pack()*.

**La funzione *premi\_btn(num)***

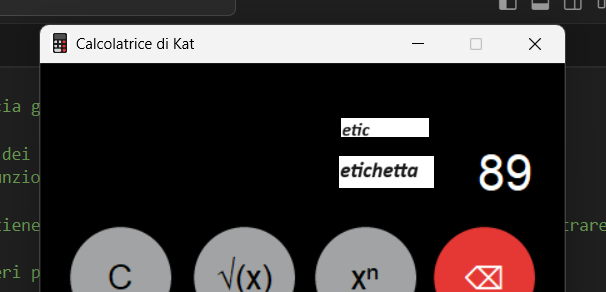
La keyword *def* comunica a Python che si sta definendo una nuova funzione. Nello specifico (da linea 3 a 8) la funzione *premi\_btn(num)* serve a trasformare l'input dei pulsanti dei numeri in testo e a mostrarlo. *num* viene definito come argomento della funzione.

Vengono definite alcune variabili come variabili globali per permettere alla funzione di manovrarle. Se non venisse specificato, il valore delle variabili muterebbe solo all’interno della funzione stessa.

Il valore di *equazione\_testo* viene aggiornato: al valore precedente di *equazione\_testo* viene aggiunto l’argomento. La stessa cosa succede per la variabile *testo*.

Per fare un esempio, quando viene premuto il pulsante del nove, “9” viene posto come argomento e viene aggiunto “9” alle variabili. Se la variabile *equazione\_testo* prima conteneva già “8”, ora *equazione\_testo* contiene “89” poiché quando si parla di stringhe, non viene fatta la somma algebrica.

L’etichetta *etichetta* viene aggiornata con il nuovo valore assunto da *testo* tramite l’apposita funzione *set()*.



**La funzione *premi\_sym(sym)***

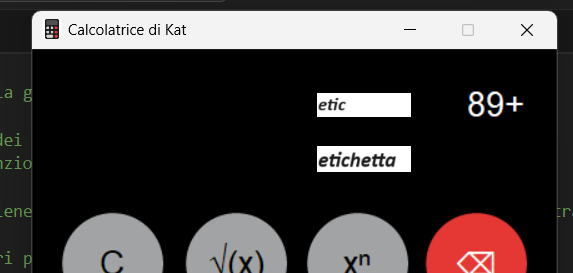
Questa funzione ha uno scopo simile alla precedente ma presenta alcune differenze. La funzione *premi\_sym(sym)* serve a trasformare l'input dei pulsanti dei simboli in testo e a mostrarlo (da linea 10 a 18). *sym* viene definito come argomento della funzione.

Anche in questo caso vengono definite alcune variabili globali.

Aggiorna il valore di *equazione\_testo* aggiungendo l’argomento al valore precedente. Vengono aggiornati di conseguenza anche i due valori dei testi. *testo* riassume un valore nullo mentre *testo\_piccolo* assume lo stesso valore di *equazione\_testo*.

Per aggiornare il testo mostrato, si fa uso della funzione *set()*: il testo in *etichetta* viene eliminato mentre il testo in *etic* diventa l’input inserito finora.

Proseguendo con l’esempio di prima, se l’input numerico è ora “89”, premendo il pulsante del più, “+” viene posto come argomento e aggiunto all’equazione. Il testo però, che sarà diventato “89+”, non verrà mostrato in *etichetta* poiché il testo in essa assume un valore nullo; verrà invece mostrato nell’etichetta più in alto, *etic*.



**La funzione *btn\_delete()***

La funzione *btn\_delete()*ha lo scopo di rimuovere l’ultimo elemento numerico che si è inserito (da linea 20 a 25). Dato che gli ultimi input numerici si trovano nella variabile *testo* allora bisognerà manipolare quest’ultima.

Per capire meglio come viene aggiornato il valore di *testo*, bisogna introdurre l’index. L’index è indicato nelle parentesi quadre dopo aver specificato la lista o la stringa a cui si riferisce. È utile per accedere agli elementi di una stringa o di una lista conoscendone la posizione. L’index degli elementi parte da 0 e procede in ordine crescente.

Prendiamo in esempio una generica lista e definiamone gli elementi indicandone l’index:

myList = [“1”, “6”, “5”, “2”]

index: 0 1 2 3

Si può accedere a un elemento anche partendo dall’ultimo. L’ultimo elemento della lista è indicato dall’index [-1], il penultimo da [-2] e così via.

È possibile anche indicare una serie di elementi compresi tra un index e un altro. Nel caso della funzione *btn\_delete()*, è necessario considerare tutti gli elementi di *testo* fino al penultimo, dove il penultimo è incluso. Per indicare gli elementi fino ad un index stabilito, si possono usare i due punti posti prima dell’index; in questo modo però l’index stabilito è escluso. Per considerare tutti gli elementi tranne l’ultimo quindi sarà necessario prendere in esame tutti gli elementi fino all’ultimo, poiché questo ne sarà escluso.

Detto tutto questo, *testo[:-1]* significa considerare tutti gli elementi di *testo* meno l’ultimo.

Per aggiornare l’equazione, si può porre uguale ai due testi insieme.

La funzione *set()* aggiorna infine il testo mostrato.

**La funzione *uguale()***

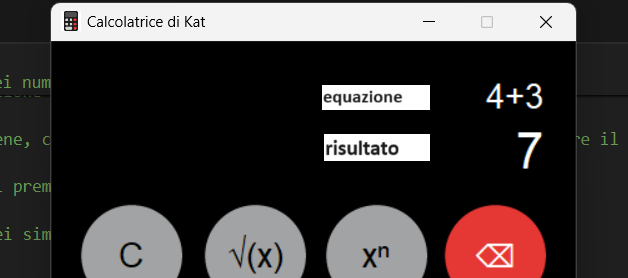
La funzione *uguale()* serve a svolgere i calcoli dopo aver premuto l’apposito pulsante dell’uguale (da linea 150 a 164).

Per evitare errori, pone come eccezione il caso in cui ci sia stato un errore di sintassi o una divisione per zero. In tal caso sostituisce il testo presente in *etichetta* con il relativo avvertimento. Poiché non altera *testo*, è possibile tornare a modificarlo ma resetta il valore di *equazione\_testo* e *testo\_piccolo*.

I calcoli vengono svolti dalla funzione *eval()*, che prende in esame il testo completo dell’equazione presente in *equazione\_testo*. Il risultato sarà contenuto nella variabile *risultato*.

Vengono rimpiazzati tramite *replace()* i caratteri che *eval()* non saprebbe gestire: questo è necessario se si vuole mostrare sullo schermo caratteri speciali o diversi da quelli che *eval()* valuta. Per esempio, sullo schermo la divisione è indicata con “÷” ma per fare il calcolo bisogna rimpiazzarlo con “/”.

Dopo aver ricavato il risultato, quest’ultimo verrà sistemato adeguatamente, per esempio con approssimazioni o riscrivendolo sotto forma di potenza in base dieci, tramite un’altra funzione. Viene richiamata la funzione *check()* che verrà trattata in seguito.



**La funzione *radice\_quadrata()***

La funzione *radice\_quadrata()* ha il fine di aggiungere al calcolo del risultato la radice quadrata dell’ultimo numero presente (da linea 166 a 180). Ha un funzionamento molto simile alla funzione *uguale()*.

Gestisce gli errori allo stesso modo della funzione precedente.

Prima di completare il calcolo con *eval()* tuttavia viene specificato di inserire alla fine dell’equazione un’elevazione a potenza. Elevare un numero alla potenza di un mezzo infatti equivale a farne la radice quadrata.

Il risultato viene strutturato nella funzione *check()* come nella funzione *uguale().*

**La funzione *rimuovi()***

La funzione *rimuovi()* cancella il testo presente nelle caselle di testo e permette di svolgere un nuovo calcolo (da linea 182 a 188).

È una funzione molto semplice che pone un valore nullo alle etichette e alle variabili di equazione e testo. Non è necessario porre un valore nullo anche a *testo\_piccolo* perché ha un valore dipendente a quello dell’equazione.

**Mostrare i pulsanti sullo schermo e renderli funzionali**

Per mostrare i pulsanti in *finestra* bisogna innanzitutto indicare la porzione di schermo in cui posizionarli. Con la funzione *frame()* si può suddividere la finestra: crea un’area rettangolare sullo schermo in cui andranno i pulsanti (da linea 210 a 212).

Al frame viene aggiunto uno sfondo nero (come quello di *finestra*) e viene posizionato sullo schermo con la funzione *pack()*.

Avere dei pulsanti di forma circolare non è possibile direttamente con Tkinter, per questo bisogna associare ai pulsanti delle immagini che rappresentano i pulsanti. Da linea 215 a 218 viene usata la classe *PhotoImage()* per associare un’immagine ad ogni variabile.

Per ogni pulsante che si vuole mostrare (linea da 221 a 268), bisogna associarne uno ad ogni variabile grazie alla classe *Button()*. Nella classe viene specificata l’appartenenza al frame, l’immagine, il testo del bottone (per esempio numeri o simboli), le caratteristiche del testo (come colore, font o dimensioni), lo sfondo e altre caratteristiche grafiche.

In *Button()* viene inoltre specificata la funzione a cui rimanda quello specifico pulsante. Per esempio, nel caso dei numeri, si associa la funzione *premi\_btn(num)* inserendo al posto dell’argomento il numero che si sta premendo. La stessa cosa avverrà per i simboli, che rimanderanno invece alla funzione *premi\_sym(sym)*. In entrambi i casi, le funzioni devono essere introdotte da “*lambda:*”: questo passaggio permette di fare riferimento alla funzione ma senza richiamarla immediatamente. Nel caso dell’uguale, della radice quadrata e di clear, non bisogna inserire “*lambda:*” prima delle funzioni correlate.

Per spiegare meglio come funzionano *premi\_btn(num)* o *premi\_sym(sym)*, si può prendere in esempio il caso del pulsante “1”. Quando si crea il pulsante con la classe *Button()*, si definisce il comando “*lambda: premi\_btn(1)*”: specificando che l’argomento è “1”, nello svolgere la funzione *num* sarà sostituito da “1”.

Per posizionare i pulsanti si fa uso della funzione *grid()*, in cui viene specificata riga e colonna, come se fosse una tabella. Per porre spazio tra i pulsanti, si specifica il padding (lo spazio tra il contenuto e il bordo di un elemento) sulle due assi cartesiane con *padx=(sinistra, destra)* o *pady=(sopra, sotto)*. Nel caso del pulsante “0”, che occupa due colonne, bisogna anche specificare il fatto che debba riempire quelle due colonne con *columnspan*.



**Ricevere input dalla tastiera**

Per poter interagire con la calcolatrice non solo con i pulsanti, ma anche con la tastiera del computer, bisogna associare ai diversi tasti le stesse funzioni che sono state associate ai pulsanti. Ciò è possibile con la funzione *bind()* in cui si deve specificare il nome del tasto della tastiera e la funzione che si vuole richiamare (da linea 271 a 294).

Questa volta tutte le funzioni devono essere precedute da “*lambda x:*”.

**La funzione *check()***

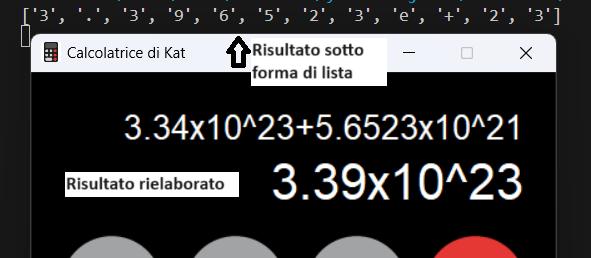
Le funzioni *uguale()* e *radice\_quadrata()* richiamano entrambi la funzione *check()* per approssimare e sistemare il risultato calcolato (da linea 27 a 148).

La prima cosa che fa questa funzione (da linea 31 a 34) è verificare la presenza dei numeri complessi. In Python la parte immaginaria dei numeri complessi è indicata con la lettera “j” quindi verificandone la presenza nel risultato si può dedurre se esso sia o non sia un numero complesso. Questo è possibile grazie all’uso del metodo *\_\_contains\_\_()*. Nel caso il risultato contenga la “j” viene mostrato il relativo avvertimento.

Grazie all’uso della costruzione “*if…else*”, se la condizione si rivela falsa, mette in pratica la parte di codice contenuta in “else:”. Avendo accertato che il risultato non è un numero complesso, procede per cercare di capire quali caratteristiche abbia il risultato con un’altra serie di condizioni.

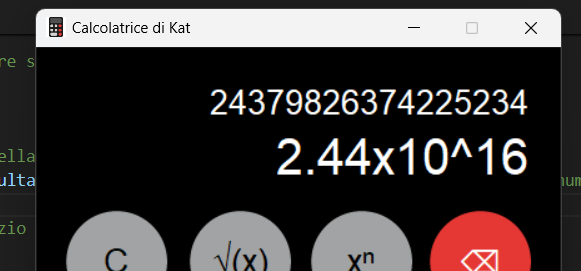
Come prossimo passaggio, controlla, a simil modo di prima, se il risultato contiene “e” (da linea 36 a 55): “e” in Python indica la potenza in base 10. Alcune volte infatti, il risultato è già scritto sotto forma di potenza in base dieci. In questo caso bisogna lavorare con le liste.

* Trasforma il risultato in una lista, necessario per manipolare gli elementi al suo interno. Questo non sarebbe possibile lavorando con una stringa (linea 37).
* Trova in che posizione si trova la “e” all’interno della lista. Ciò è possibile attraverso la funzione *index()*. Conoscendo dove si trova la “e”, si sostituisce con “x10^” per renderlo coerente al modo in cui la calcolatrice mostra il testo (linea 38 e 39).
* Poiché Python riscrive le potenze di 10 che hanno una sola cifra mettendo lo zero davanti (come per esempio 1.2e+02), lo zero si può rimuovere sempre per coerenza. Prima di procedere, controlla se il penultimo numero (*risultato[-2]*)sia uguale a 0 e se dalla potenza in poi (*risultato[potenza:]*) ci siano quattro cifre. L’ultimo controllo serve ad evitare che venga tolto lo zero dove non è necessario, come per esempio in 4.35e+208. Nel caso le condizioni siano vere, lo zero viene eliminato tramite la funzione *pop(index)* (linea 40 e 41).
* Controllando il posizionamento di “e”, si può capire quanti decimali abbia il risultato. In caso ci siano meno di due decimali, aggiunge degli zero per coerenza in modo che ci sia un risultato del tipo 1.00x10^n. L’obiettivo viene raggiunto tramite la funzione *insert(index, object)* (da linea 42 a 47).
* Tramite la funzione *len()* si può ricavare quanti elementi ci sono in una determinata lista. Nel caso di un numero positivo, il primo decimale di troverà all’index [2] mentre l’ultimo, avrà come index lo stesso della potenza ma diminuito di uno. L’obiettivo è eliminare eventuali decimali dopo il secondo: dopo aver controllato che effettivamente il numero sia positivo (quindi che non abbia un “-” all’inizio) e che gli elementi tra il secondo decimale e la potenza siano più di due, elimina tutti gli elementi dal terzo decimale all’ultimo. Nel caso di un numero negativo, l’index da cui partono i decimali sarà aumentato di uno e bisogna intervenire di conseguenza (da linea 48 a 51).
* Per evitare che la potenza mostri il segno “+” nel risultato (come in 1.23x10^+12), controlla che la lista contenga quel carattere e nel caso lo elimina. (da linea 52 a 54)
* Nell’ultimo passaggio converte la lista in una stringa (linea 55).



Se non si sta considerando un numero che sia già scritto come potenza di 10, si filtra per vedere se invece sia un numero maggiore o uguale a 100000000 oppure minore o uguale a -100000000. Si pongono entrambe le condizioni per considerare sia numeri positivi che negativi (da linea 56 a 86).

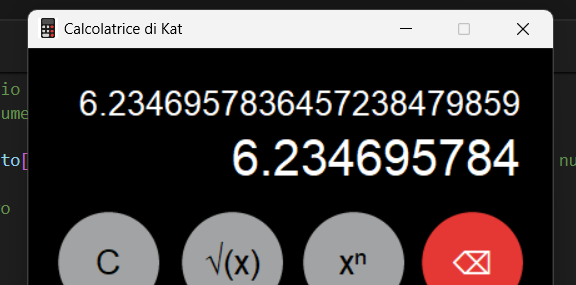
* La prima distinzione che viene fatta serve a rimuovere i decimali dato che per riscrivere il risultato sotto forma di potenza in base dieci non saranno necessari. Nel caso di un numero intero scritto nella forma n.0, tiene conto solo degli elementi nella lista che siano presenti prima del punto. Nel caso di un numero decimale, elimina la parte decimale trovando l’index del punto (da linea 57 a 64).
* Per integrare sia numeri positivi che negativi, elimina il meno nel caso esso sia presente nel risultato. La variabile *numero* serve a tener conto se il numero iniziale fosse negativo o positivo (da linea 65 a 68).
* Il prossimo passaggio ha la funzione di approssimare il risultato. Controlla se la quarta cifra del numero iniziale sia maggiore di cinque, in tal caso procede. Controlla se i numeri da approssimare siano diversi da nove (“diverso da” in Python è indicato con !=), in caso contrario agisce di conseguenza. Approssima aumentando di uno il valore della cifra da approssimare (da linea 69 a 83).
* Dopo aver manipolato gli elementi della lista a dovere, per riscrivere il risultato accede a quegli elementi e li usa per creare un nuovo risultato sotto forma di stringa. Il risultato è riscritto aggiungendo un punto tra i primi due elementi della lista, aggiungendo “x10^” dopo il terzo elemento e contando di quante cifre è composto il risultato, diminuendo di uno per non considerare la prima cifra. Se il numero iniziale era negativo, aggiunge un meno all’inizio del risultato (da linea 84 a 86).



Il prossimo tipo di numeri che considera sono i numeri interi che non fanno parte delle prime due categorie. In questo caso non apporta modifiche al risultato (linea 87 e 88).

Avendo scartato la possibilità che il risultato sia un numero intero, il risultato è per forza un decimale. Nel caso di un decimale maggiore di 1 o minore di -1 che abbia più di dieci cifre, rielabora a tal modo il risultato (da linea 89 a 102):

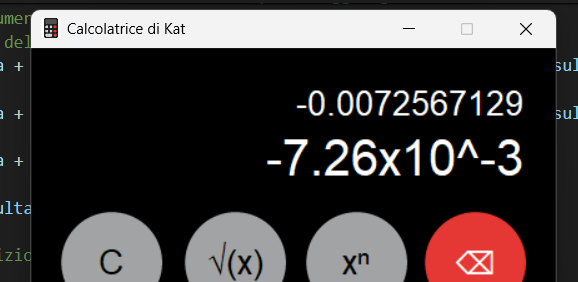
* Per prima cosa, rimuove il meno nel caso di un numero negativo e modifica la variabile *numero* per tener conto del fatto che il numero iniziale fosse negativo (da linea 91 a 95).
* Approssima il risultato nel caso in cui la cifra da approssimare sia diversa da nove e quella dopo sia maggiore di cinque. (da linea 96 a 99)
* Infine tronca la lista fino all’undicesimo elemento e nel caso di un numero iniziale negativo aggiunge un meno all’inizio (da linea 100 a 102).



Nel caso si tratti di un risultato di modulo compreso tra 0.95 e 1, lo approssima direttamente a 1 (da linea 103 a 106).

Nella circostanza in cui il risultato sia un numero compreso tra 1 e -1, viene riscritto come potenza negativa in base dieci (da linea 107 a 144).

* Controlla se il numero è negativo, se così fosse rimuove il meno e aggiorna la variabile *numero* (da linea 109 a 112).
* Rimuove i primi due elementi della lista (quindi elimina “0.”) poiché sono superflui nel riscrivere il risultato sotto forma di potenza (linea 113).
* Con il comando *next()* trova l’index della prima cifra diversa da zero attraverso a un loop. Passa al prossimo index della lista se trova uno zero. L’index trovato sarà custodito nella variabile *cifra* (linea 114).
* Dopo essersi assicurato che dopo *cifra* ci siano almeno altre quattro cifre, elimina dalla quarta cifra dopo la prima diversa da zero in poi (linea 115 e 116).
* Se sono presenti esattamente quattro cifre da *cifra* in poi, approssima il risultato nel caso in cui la quarta dovesse essere maggiore di cinque, considerando le evenienze in cui le cifre prima di quella da approssimare siano uguali a nove (da linea 117 a 135).
* Riscrive il risultato tenendo conto degli elementi della lista e adattandosi nel caso non ci siano abbastanza elementi dopo la prima cifra diversa da zero (da linea 136 a 142).
* Se il numero iniziale era un numero negativo, inserisce un meno all’inizio (linea 143 e 144).



Le ultime quattro linee della funzione (da linea 145 a 148), sono comuni a tutti i casi in cui il risultato non sia costituito da una parte immaginaria. Queste linee di codice hanno lo scopo di mostrare il risultato sullo schermo, resettare il valore del piccolo testo in alto e porre la nuova equazione uguale al risultato per poter partire da esso nello svolgere un nuovo calcolo.