import turtle

create a Turtle from turtle and name it calculator

create a Screen from turtle named screen

call turtle.tracer with (0, 0)

hide the cursor

keep the pen up

Set a constant variable SIZE and make it equal to 50

CLASS SCREEN PSEUDOCODE:

Create a class called **Screen**:

Initiate class variables (pass in: x, y, width, height, numLines):

Set **self.x** to value x

Set **self.y** to value y

Set **self.width** to value width

Set **self.fontSize** to 10

Set **self.numLines** to value numLines

Set **self.lineHeight** to height divided by self.numLines (integer division)

Set **self.height** to self.lineHeight multiplied by self.numLines

Set **self.lines** to an empty array with the size of self.numLines

Define a method called **reset** (pass in: no arguments):

Set **self.fontSize** to 10

Set **self.lines** to an empty array with the size of self.numLines

Define a method called **add** (pass in: value):

if the texts no longer fit on the calculator screen:

if self.fontSize is greater than 5:

Subtract 1 from self.fontSize

else:

Do not add another number to the screen

if self.lines[1] has nothing in it and value is a digit:

set **self.lines[0]** to an empty string

else if self.lines[1] has nothing and the value is not a digit:

set **self.lines[1]** to self.lines[0]

Define a method called **evaluate** (pass in: no arguments):

Set **self.lines[0]** to self.lines[1]

Set **self.line[1]** to an empty string

Define a method called **display** (pass in: no arguments):

Tell turtle to go to position (self.x, self.y)

Tell turtle to put pen down

for 2 times:

Tell turtle to go forward for self.width pixels

Tell turtle to turn left 90 degrees

Tell turtle to go forward for self.height pixels

Tell turtle to turn left 90 degrees

Tell turtle to put pen up

for every line in **self.lines**:

Set **lineLen** to the length of the line

for every character in the current line:

Go to position of where the digit should appear on the screen

Display (write) the current character to the screen

CLASS BUTTON

Create a class called **Button**:

Initiate the class (pass in: x, y, value):

Set self.x to value x

Set self.y to value y

Set self.value to value value

Set self.size to SIZE

Define a method called clicked (pass in: xPos, yPos):

if xPos is in between self.x and self.x+self.size and yPos is in between self.y and self.y+self.size:

return True

Define a method called display (pass in: no arguments):

Go to position (self.x, self.y)

Put the pen down

#Draw the border of a square

for 4 times:

Move forward for self.size pixels

Turn left 90 degrees

Lift the pen up

Go to the middle of the square

Display (write) self.value with center alignment, font (Arial, 30, normal)

FUNCTIONS

Define a function called addToScreen (pass in: value, screen):

if value is '=':

Call screen's evaluate method

else:

Add the value to the screen

Define a function called evaluate (pass in: line):

Set closedBracket to True

For each character in the line:

if the character is '(':

Set closedBracket to False

else if the character is ')':

Set closedBracket to True

If the bracket is closed (closeBracket) and the last character of the line isn't an operator:

return the evaluated line

else return 'Error'

Define a function called check (pass in: x, y):

Clear the entire screen

for every button in the list buttons:

if the button is clicked:

if it's the clear button:

Reset the calculator screen

else:

Add the value to the screen

call the displayAll function to display everything

Define a function called displayAll (pass in: no arguments):

for every button in the list buttons:

call the button's display method

call the calculator screen's display method

CREATING THE BUTTON LIST

Initiate an empty buttons list

# Append Button objects to the buttons list

for i in range(3):

for j in range(1, 4):

append Button objects to buttons with (x = i\*SIZE, y = j\*SIZE, value = i+(j-1)\*3+1)

append a Button 0 at (0, 0)

append a Button . at (SIZE, 0)

append a Button = at (2\*SIZE, 0)

append a Button / at (3\*SIZE, 0)

append a Button \* at (3\*SIZE, SIZE)

append a Button - at (3\*SIZE, 2\*SIZE)

append a Button + at (3\*SIZE, 3\*SIZE)

append a Button C at (-SIZE, 3\*SIZE)

append a Button ( at (-SIZE, 2\*SIZE)

append a Button ) at (-SIZE, SIZE)

Create a Screen called calcScreen

call the displayAll function to display everything

create an infinite loop for the game

constantly check the screen for clicks

constantly update the turtle (display screen)