Useful websites
TI-84 Plus/TI-84 Plus Silver Edition Guidebook,
https://education.ti.com/en/guidebook/details/en/C4D11EB6D86B47D19CD768E54A967441/84p
TI-84 Connect CE Software, https://education.ti.com/software/search/ti-84-plus-family-ti-83-plus-family
TI-84 Apps, https://education.ti.com/software/search/ti-84-plus-family-ti-83-plus-family
TI-84 Programs, https://www.ticalc.org/pub/83plus/basic/math/
TI Program Editor, https://www.cemetech.net/sc/
TI-83/84/89/92 Procedures and Help, https://brownmath.com/ti83/
Calculus Using the TI-84, https://education.ti.com/html/t3_free_courses/calculus84_online/index.html

Command catalog:
a.) [2ND] [CATALOG]
b.) press letter to access the Catalog section that begins with the pressed letter
c.) scroll down to access a command
d.) [ENTER]

Catalog help:
a.) download from CtlgHelp app from https://education.ti.com/software/search/ti-84-plus-family-ti-83-plus-family
b.) transfer from computer to calculator using cable and TI Connect CE software
c.) [APPS] CtlgHelp
d.) [2ND] [CATALOG]
e.) scroll to catalog entry
f.) [+]

Useful Ti-84 apps:
Finance – solves financial math problems
CellSheet – spreadsheet that simulates Excel
Conics – graph conic equation
CtlgHelp – provides function parameters
Periodic – periodic table
PlySmlt2 –polynomial root finder and simultaneous equation solver
SciTools – sig figs calculator, unit converter, data grapher, vector calculator

Lowercase letters: https://www.youtube.com/watch?v=Oac7x_uDnKM

Graphing a piecewise function:
a.) enter each piece of the function in { } followed by the interval over which it is defined
b.) function pieces must be separated with +
c.) function pieces must be enclosed in { and }
d.) domains with two endpoints must be entered separately
e.) [2ND] [TEST] to enter an =, ≥, >, ≤, or <

Example: $f(x) = \begin{cases} x+4, x < -1 \\ 3x-2, -1 \le x < 2 \\ 2x+3, 2 \le x \end{cases}$

[Y=] enter the equation: {X + 4}{X < -1} + {3X – 2}{-1 ≤ X}{X < 2}+ {2X + 3}{2 ≤ X}
in this example, the second function piece's domain has beginning and ending points so must be entered as
{-1 ≤ X}{X < 2}
[GRAPH]

Graphing a function, its derivative, and a tangent line at point P:
a.) [Y=] enter the function into Y1
b.) [Y=] [↓] [MATH] [8:nDeriv(] [X] [VARS] [→] [1:Function] [1:Y1] [→] [X] [ENTER] to enter the derivative of Y1 into Y2
c.) [2ND] [QUIT]
d.) [GRAPH] to graph the function and its derivative
e.) [2ND] [QUIT]
f.) [2ND] [DRAW] [5:Tangent(] [VARS] [→] [1:Function] [1:Y1] [,] [P] (where P is an x-coordinate of Y1) [ )] [ENTER] to graph the tangent line of point P

Viewing the function graph and its table simultaneously:
a.) [Y=] enter the function [2ND] [QUIT]
b.) [MODE]
c.) [↓] (7 times) to highlight FULL
d.) [→] [→] to highlight G-T
e.) [ENTER]
f.) [GRAPH]
g.) [2ND] [TABLE] to cycle through the X and Y values
h.) [2ND] [QUIT] to exit

Note: In the following programs, spaces, indents, and blank lines are included for clarity and are not part of the program.

Note: The use of lowercase letters is optional EXCEPT for those used in Statistics programs.

Note: To enter variables into programs use the following:
Window variables: [VARS] [1] [select variable]
Statistics variables: [VARS] [5] [select variable]
String variables: [VARS] [7] [select variable]
Function variables: [VARS] [→] [1] [select variable]
Parametric: [VARS] [→] [2] [select variable]

Note: TI-84 calculators will occasionally display "INVALID DIM" error. To fix: [2^{nd}] [STATPLOTS] [4:PLOTSOFF]

TI-84 Calculator Programs

Name Program – display name, email address, time, date, press [ENTER] twice to exit this program	Explanation
AxesOff	turn axes off [2ND] [CATALOG] [A] scroll down to AxesOff [ENTER]
ClrDraw	clear graph window [2ND] [DRAW] [1:ClrDraw]
Text(15, 28, "enter your name here")	display name [2ND] [DRAW] [0:Text(]
Text(25, 19, "enter first part of your email address here")	display first part of email address [2ND] [DRAW] [0:Text(]
Text(32, 26, "enter second part of your email address here")	display second part of email address [2ND] [DRAW] [0:Text(]
0 → K	store 0 to K
While K = 0	while check key press [PRGM] [5:While]
getTime → L_1	store time (hours, minutes, seconds) in list L_1 [2ND] [CATALOG] [G] scroll down to getTime [ENTER]
getDate → L_2	store date (year, month, day) in list L_2 [2ND] [CATALOG] [G] scroll down to getDate [ENTER]
":" → Str1	store colon to Str1 [VARS] [7] [1:Str1]
":" → Str2	store colon to Str2 [VARS] [7] [2:Str2]
If $L_1(2) < 10$	if minutes < 10 [2ND] [TEST] [5:<]
Then	then
":0" → Str1	pad minutes with colon and leading 0 [VARS] [7] [1:Str1]
End	end check minutes [PRGM] [7:End]
If $L_1(3) < 10$	if seconds < 10 [PRGM] [1:If] [2ND] [TEST] [5:<]
Then	then [PRGM] [2:Then]
":0" → Str2	pad seconds with colon and leading 0 [VARS] [7] [2:Str2]
End	end check seconds [PRGM] [7:End]
Text(42, 12, $L_1(1)$, Str1, $L_1(2)$, Str2, $L_1(3)$)	display hours, minutes, seconds [2ND] [DRAW] [0:Text]

	[VARS] [7] [1:Str1] [VARS] [7] [2:Str2] [VARS] [7] [3:Str3]
"/" → Str3	store slash to Str3
If $L_2(3) < 10$	if day < 10 [2ND] [TEST] [5:<]
Then	then [PRGM] [2:Then]
"/0" → Str3	pad day with slash and leading 0 [VARS] [7] [3:Str3]
End	end [PRGM] [7:End]
Text(42, 50, $L_2(2)$, Str3, $L_2(3)$, "/", $L_2(1)$)	display month, day, year [2ND] [DRAW] [0:Text]
getKey → K	get key press [PRGM] [→] [7:getKey]
End	end while key press [PRGM] [7:End]
Pause	pause [PRGM] [8:Pause]
ClrDraw	clear graph window [2ND] [DRAW] [1:ClrDraw]

Quadratic Program – vertex, intercepts, graphs equation	Explanation
a+b*i*	set to complex mode
Disp "$y = Ax^2 + Bx + C$"	display quadratic equation form
Prompt A, B, C	enter coefficients
$B^2 - 4AC \rightarrow D$	discriminant (before square root calculated)
$2A \rightarrow E$	quadratic formula denominator
" "→Str4	store space in Str4
If D < 0	if discriminant < 0
Then	then
-D → D	make discriminant positive
"*i*" → Str4	store *i* to Str4
End	end check if discriminant < 0
(-B + √(D)) / E → F	first x-value root
(-B - √(D)) / E → G	second x-value root
-B / E → H	vertex x-coordinate
$AH^2 + BH + C \rightarrow I$	vertex y-coordinate
H – 1 → J	a second x-coordinate
H + 1 → L	a third x-coordinate
$AJ^2 + BJ + C \rightarrow K$	a second y-coordinate
$AL^2 + BL + C \rightarrow M$	a third y-coordinate
{H, J, L} → L_1	store x-coordinates in list L_1

{I, K, M} → L_2	store y-coordinates in list L_2
QuadReg Y_1	calculate quadratic equation
Equ▸String(Y_1, Str1)	convert quadratic equation to a string
"+" → Str2	store + to String2
"+" → Str3	store + to String3
If B < 0	if x-coefficient < 0
Then	then
"-" → Str2	store – in String2
End	end check if x-coefficient < 0
If C < 0	if constant < 0
Then	then
"-" → Str3	store – in String3
End	end check if constant < 0
H – 10 → Xmin	calculate Xmin
H + 10 → Xmax	calculate Xmax
I – 10 → Ymin	calculate Ymin
I + 10 → Ymax	calculate Ymax
ClrDraw	clear graph window
ZStandard	set up standard axes
AxesOn	turn axes on
LabelOff	turn axis labels off
1 → R	initiailize display row counter
9 → S	initialize display row increment
Text(R, 2, "y=", round(A, 2), "x^2", Str2, abs(round(B, 2)), "x", Str3, abs(round(C, 2)))	display equation with correct signs between terms
R + S → R	increment display row counter
If D ≥ 0	if discriminant ≥ 0
Then	then
Text(R, 2, "(", round(F, 2), ",0) , (", round(G, 2), ", 0) , (0, ", round(C, 2), ")")	display x-intercepts and y-intercept
Else	else
Text(R, 2, (0, ", round(C, 2), ")")	display y-intercept (no x-intercepts)
End	end discriminant ≥ 0
R + S → R	increment display row counter
Text(R, 2, "x=(", -round(B, 2), "+-√(", abs(round(D, 2)), ")", Str4, ")/", round(E, 2))	display quadratic equation solution formula
R + S → R	increment display row counter
1 → W	initialize factor outside of discriminant
If D ≠ 0	if discriminant ≠ 0

Then	then
For (Q, 1, 10)	repeat this loop 10 times (10 is arbitrary)
For (T, 2, 20)	repeat loop 20 times (20 is arbitrary but should be large enough to provide squared factors of discriminant)
If not(fPart (D / T^2))	if discriminant is divisible by T^2
Then	then
D / T^2 → D	divide discriminant by T^2 and store result in discriminant D
W * T → W	factor outside discriminant by T and store in W
End	end check if T^2 is factor of discriminant
End	end for T loop
End	end for Q loop
End	end discriminant ≠ 0
If W≠ 1 and D ≠ 1	if square-rooted discriminant factors ≠ 1 and discriminant ≠ 1, discriminant ≠ perfect square
Then	then
Text(R, 7, "=(", -round(B, 2), "+-", W, "√(", abs(round(D, 2)), ")", Str4, ")/", round(E, 2))	display solution with radical symbol
R + S → R	increment display row counter
End	end if simplified discriminant
If W ≠ 1 and D = 1	if discriminant = perfect square
Then	then
Text(R, 7, "=(", -round(B, 2), "+-", W, Str4, ")/", round(E, 3))	display solution without radical symbol
R + S → R	increment display row counter
End	end if discriminant is a perfect square
If D ≥ 0	if discriminant ≥ 0
Then	then
Text (R, 7, "=", round(F, 2), Str4, ",", round(G, 2), Str4)	display decimal values of x
Else	else
Text (R, 7, "=", round(real(F), 2), "+", abs(round(imag(F), 2)), "*i*,", "", round(real(G), 2), "-", abs(round(imag(G), 2)), "*i*,")	display complex values of x
End	end discriminant ≥ 0
R + S → R	increment the display row counter
Text (R, 2, "Vertex (x, y) = (", round(H, 2), ",", round(I, 2), ")")	display the vertex
Pause	pause
Lbl 0	end of program

Factors Program – prime factors of positive integer	Explanation
Fix 0	set number of displayed decimal places to zero
ClrDraw	clear graph window
AxesOff	turn axes off
FnOff	turn functions off
ZStandard	set up standard drawing window
ClrList L_1	clear list L1
Input "Integer:", N	enter a number for which prime factors will be calculated
iPart(abs(N)) → N	entered number converted into a positive integer
1 → I	set list L_1 index at 1
For (J, 2, √(N))	set up loop that stops at the square root of input number N
N / J → T	divide input number N by loop counter and store in T
If T = int(T)	if T is an integer
Then	then
N / T →L_1(I)	store input number N/T in list L_1, N/T is a factor of N
T → L_1(I+1)	store T in the next location in list L_1, T is the factor of N that pairs with N/T
I+2 → I	increment the list counter by 2 for the next pair of factors of N
End	end if T is an integer
End	end factor of N For loop
If I = 1	if I is still 1
Then	then
Disp "Prime"	display that the input number is prime
Pause	pause
End	end if the number is prime statement
I – 2 →I	subtract 2 from I
1 → L	store 1 in L, the display column number
32 → M	store 32 in M, the display column increment
2 → D	store 2 in D, the display row number
8 → P	store 8 in P, the display row increment
L → C	store L in C, start the column number at L
D → R	store D in R, start the row number at D
Text(R, C, "Factors of ", N, ":")	display factors of input number
R + P → R	increment the display row number by 8
For(K, 1, I, 2)	start display factors of N loop
Text(R, C, L_1(K), "," , L_1(K+1))	display factors of N at row R, column C

C + M → C	increment display column number by M
If C > 80	if display column number > 80
Then	then
L → C	reset the display column number counter
R + P → R	increment the display row number counter
End	end display column check If loop
If R > 55	if the display row number counter > 55
Then	then
D → R	reset display row number counter
Pause	pause
ClrDraw	clear graph window
End	end display row check If loop
End	end display factors of N loop
Pause	pause
ClrList L_1	clear list 1
Float	turn float mode on
FnOn	turn functions on

Graph Piecewise Function Program (maximum of 4 pieces; enter domains in the form of C, X oper C, or C1 oper X oper C2, where C, C1, and C2 are constants, and oper is =, >, ≥, <, ≤	Explanation
" " → Str1	store space in Str1
" " → Str2	store space in Str2
" " → Str3	store space in Str3
" " → Str4	store space in Str4
" " → Str7	store space in Str7
Input "Number of pieces (2-4):", N	enter number of function pieces between 2 and 4
int(abs(N)) → N	convert number of function pieces to positive integer
If N < 2 or N > 4	if number of function pieces < 2 or > 4
Then	then
2 → N	set number of pieces at 2
End	end number of function pieces
ClrDraw	clear graph window
AxesOn	turn axes on
Input "Xmin:", Xmin	enter Xmin
Input "Xmax:", Xmax	enter Xmax
Input "Ymin:", Ymin	enter Ymin
Input "Ymax:", Ymax	enter Ymax
For (I, 1, N)	set up input function pieces loop
Input "function:", Str8	input function piece
Input "domain:", Str9	input function piece domain

If I = 1	if 1st function piece
Then	then
"function: " + Str8 + " domain:" + Str9 → Str1	store 1st function and domain in Str1
End	end 1st function piece
If I = 2	if 2nd function piece
Then	then
"function: " + Str8 + " domain:" + Str9 → Str2	store 2nd function and domain in Str2
End	end 2nd function
If I = 3	if 3rd f unction
Then	then
"function: " + Str8 + " domain:" + Str9 → Str3	store 3rd function and domain in Str3
End	end 3rd function
If I = 4	if 4th function
Then	then
"function: " + Str8 + " domain:" + Str9 → Str4	store 4th function and domain in Str4
End	end 4th function
"+{" + Str8 + "}" → Str8	put brackets around function piece
length(Str9) → L	calculate length of domain string
inString(Str9, "X") → V	find location of X variable in domain string
If V = 0	if X variable not contained in domain string, domain is a constant
Then	then
"{" + Str9 + "}" → Str9	put brackets around domain string
End	end no X variable in domain string, domain is constant
If V ≠ 0 and V ≠ 1 and V ≠ L	if X variable embedded within domain string
Then	then
"{" + subs(Str9, 1, V) + "}{" + subs(Str9, V, L – V + 1) + "}" → Str9	disassemble domain string and put brackets around each part of domain string
End	end input function pieces
Str7 + Str8 + Str9 → Str7	add function and domain string to end of entire function and domain string
End	end
length(Str7) → L	store length of function and domain string in L
subs(Str7, 3, L - 2) → Str7	remove space and + from beginning of function string
Str7 → Y1	store function and domain string in Y1
Text(49, 2, Str1)	display 1st function and domain
Text(57, 2, Str2)	display 2nd function and domain
If N ≥ 3	if number of function pieces ≥ 3
Then	then

Text(49, 55, Str3)	display 3rd function and domain
End	end number of functions pieces ≥ 3
If N = 4	if number of function pieces = 4
Then	then
Text(57, 55, Str4)	display 4th function and domain
End	end number of function pieces = 4
Pause	pause

Base Conversion Program - change of base conversion for bases 2 to 20	
Lbl 1	start program
ClrHome	clear home screen
AxesOff	turn axes off
Input "Num:", Str1	input number to be converted
Input "In base:", P	input base of input number
Input "Out base:", Q	input base of output number
If P > 20 or P < 2 or Q > 20 or Q < 2	if either input base > 20 or input base < 2
Then	then
Disp "base≠[2,20]"	display error message
Pause	pause
Goto 1	go to beginning of program
End	end base check
"0123456789ABCDEFGHIJ" → Str3	store valid character string in Str3
length(Str1) → D	store digit length of input number in D
D → dim(L_1)	dimension list L_1 to D characters
For (I, 1, D)	set up loop for number of digits in input number
sub(Str1, I, 1) →Str2	store input number digit in Str2
If inString(Str3, Str2) = 0	if input number digit not in valid digit list
Then	then
Disp "Inval num"	display error message
Pause	pause
Goto 1	go to beginning of program
End	end check input number digits
inString(Str3, Str2) – 1 → L_1(I)	store numeric location -1 of input digit in valid digit list to list L_1 (1 is subtracted because string digit locations begin with 1 not 0)
End	end input number digit counter loop
For (I, 1, D)	set up loop for number of digits in input number
If L_1(I) > P	if input number digit value > input base
Then	then
Disp "Num wrong base"	display error message
Pause	pause

Goto 1	go to beginning of program
End	end input digit value check
End	end input number digit counter
0 → T	initialize base 10 total to T
For (I, 1, D)	set up loop for number of digits D in input number
L1(I) * P^(D – I) + T → T	multiply input number digit by input base^(exponent for that location of input number) and increment total
End	end input number digit counter
T → U	store base 10 value of input number in U
If U > 1000000000	if base 10 value > 1,000,000,000
Then	then
Disp "Num > 10^9"	display error message
Pause	pause
Goto 1	go to beginning of program
End	end check base 10 value
For (I, 40, 1, -1)	set up loop to determine highest power of output base that can divide into base 10
If T / (Q^(I – 1)) ≥ 1	if base 10 value T is divisible by power of output base Q (subtract 1 to account for 1s place)
Then	then
I → dim(L_2)	dimension list L_2 to highest power
I – 1 → E	store highest power to E
Goto 2	go to 2
End	end determine highest power of output base that can divide base 10 value
End	end loop to determine highest power
Lbl 2	label 2
For (I, E, 0, -1)	set up loop to decrement exponent values
int(T / (Q^I)) → L_2(I + 1)	determine number of times power of output base divides into base 10 number and store in list L_2
T – int(T / (Q^I)) * (Q^I) → T	subtract number of times power of output base divides into base 10 number T and store in T
End	end decrement exponent values loop
ClrDraw	clear graph window
Text(2, 2, "Input=")	display Input =
Text(2, 40, Str1)	display input number
Text(10, 2, "In base=")	display Input base =
Text(10, 40, P)	display input base value
Text(18, 2, "Dec=")	display Dec =
Text(18, 40, U)	display decimal value of input number
Text(26, 2, "Out base=")	display Out base =

Text(26, 40, Q)	display output
Text(34, 2, "Output=")	display Output =
"ABCDEFGHIJ" → Str4	store valid alphabetic characters to Str4
34 → R	store 34 to row counter
40 → C	store 40 to column counter
For (I, E, 0, -1)	set up exponent values loop
If $L_2((I + 1) \geq 10$	if list L_2 digit value ≥ 10
Then	then
$L_2(I + 1) - 9$ → H	subtract 9 from list L_2 digit value since digit needs to be replaced with a letter
sub(Str4, H, 1) → Str5	replace digit value with letter value
Text(R, C, Str5)	display letter digit
Else	else
Text(R, C, $L_2(I + 1)$)	display numeric digit
End	end exponent values loop
C + 4 → C	increment column display counter
If C ≥ 80	if column display counter > 80
Then	then
R + 8 → R	increment row display counter
40 → C	reset column display counter
End	end check column display counter
End	end exponent values loop
Pause	program done

Inverse Program – display a function and its inverse	Explanation
Input "Function:", Str1	input function in Str1
Str1 → Y_1	store function in Y_1
ClrDraw	clear graph window
DispGraph	display equation graph
DrawInv Y_1	display inverse of Y_1
Text(2, 10, "Function: ", Str1)	display function
Pause	pause

Normal Distribution Program – normal distribution area, z-scores	Explanation
Input "Lower raw score:", L	input lower raw score
Input "Upper raw score:", U	input upper raw score
Input "Mean:", M	input mean
Input "Std dev:", S	input standard deviation
If S < 0	if standard deviation S < 0
Then	then
-S → S	convert standard deviation S to positive number
End	end check standard deviation

If L > U	if lower score > upper score
Then	then
Disp "Invalid input"	display invalid input
Pause	pause
Goto 0	end program
End	end
M – 3S → Xmin	store mean – 3 std devs in Xmin
M + 3S → Xmax	store mean + 3 std devs in Xmax
0 → Ymin	store 0 to Ymin
normalpdf (M, M, S) → Ymax	store y value of mean M into Ymax
normalcdf (L, U, M, S) → A	calculate area under normal distribution curve between lower raw score and upper raw score
ShadeNorm (L, U, M, S)	shade area under curve between L and U
Text(49, 30, "Area=", round(A, 4))	display shaded area
Text(1, 4, "L=", L)	display lower score L
Text(8, 4, "U=", U)	display upper score U
Text(15, 4, "Mean=", M)	display mean M
Text(22, 4, "Std dev=", S)	display standard deviation S
Text(1, 58, "zmin=", round((L – M) / S, 3))	display z-score for lower raw score
Text(8, 58, "zmax=", round((U – M) / S, 3))	display z-score for upper raw score
Pause	pause
Lbl 0	end program

Statistical Plots Program – display dot and box plots and histograms for input data	Explanation
ClrAllLists	clear all lists
FnOff	turn functions off
Input "Number of data points:", N	input number of data points
For (I, 1, N)	start data point entry loop
Input P	enter data point
P → L_1(I)	store data point values in list L_1 (x-coordinates)
End	end data point entry loop
SortA(L_1)	sort list L_1
0.2 →L_2(1)	store 0.2 to L_2(1), the y-coordinates of the data points
For (I, 2, N)	start data point y-coordinate calculation loop
If L_1(I) = L_1(I-1)	if values of two adjacent data points are equal
Then	then
L_2(I-1) + 0.2 → L_2	increment y-coordinate of data point
Else	else
0.2 → L_2(I)	store 0.2 to L_2(I)
End	end if x-values of adjacent data points are equal

End	end data point y-coordinate calculation loop
0 → Xmin	set up Xmin
max(L_1) + 1 → Xmax	set up Xmax
0 → Ymin	set up Ymin
max(L_2) / 0.2 → Ymax	set up Ymax
Lbl 5	set up plots
PlotsOff	turn plots off
ClrDraw	clear graph window
ClrHome	clear home screen
Menu("PLOT", "Dot", 1, "Box", 2, "Histogram", 3, "End", 4)	select plot type
Lb1 1	dot plot routine
AxesOn	turn axes on
Plot1(Scatter, L_1, L_2, ▫)	setup scatter plot
Goto 9	go to display plot
Lbl 2	box plot routine
AxesOff	turn axes off
1-Var Stats	calculate one variable statistics
Plot1(Boxplot, L_1)	setup box plot
Text(18, 2, "Min=", min(L_1))	display minimum value
Text(26, 2, "Q1=", Q1)	display first quartile value
Text(34, 2, "Med=", Med)	display median value
Text(42, 2, "Q3=", Q3)	display third quartile value
Text(50, 2, "Max=", max(L_1))	display maximum value
Goto 9	go to display plot
Lbl 3	histogram routine
AxesOn	turn axes on
ZoomStat	setup graph window
Input "Bin size:", B	input histogram bin size in B
B → Xscl	store histogram bin size B to Xscle
0 → Xmin	store 0 to Xmin
(iPart(max(L_1)/B)+1)*B → Xmax	set Xmax at to right-hand value of highest bin
0 → Ymin	store 0 to Ymin
Input "Ymax:", M	enter Ymax into M
M → Ymax	store M to Ymax
Plot1(Histogram, L_1)	setup histogram
Goto 9	go to display plot
Lb1 9	display plot routine
DispGraph	display dot plot/box plot/histogram
Pause	pause

FnOn	turn functions on
Goto 5	select a different plot using the same data
Lbl 4	end

Tangent Line Program – display function and tangent line at a given point, display equation of tangent line	Explanation
AxesOn	turn axes on
Input "Equation:", Str1	input equation
Str1 → Y1	store string in Str1 into Y1
Input "X-value:", P	input x-coordinate of point at which to calculate tangent line
nDeriv(Y1, X, P) → M	calculate slope M of tangent line at P and store in M
P → X	store x-coordinate in X
Y1 → Q	store y-coordinate of Y1 calculated at X in Q
Str1 → Y1	reset Y1
M * -P + Q →	calculate y-intercept of tangent line
" " → Str2	initialize Str2
If B > 0	if tangent line intercept > 0
Then	then
"+" → Str2	y-intercept is positive, store + to Str2
Else	else
If B < 0	if tangent line intercept < 0
Then	then
"-" → Str2	y-intercent is negative, store – to Str2
End	end tangent line intercept < 0
End	end tangent line intercept > 0
P – 10 → Xmin	calculate Xmin
P + 10 → Xmax	calculate Xmax
Q – 10 → Ymin	calculate Ymin
Q + 10 → Ymax	calculate Ymax
DispGraph	display equation graph
Tangent(Y1, P)	display tangent line at X = P
Text(8, 8, "Equation: Y= ", Str1)	display equation
Text(29, 8, "(X, Y) = (", P, ", ", Q,")")	display point coordinates for tangent line
If M ≠ 0 and B ≠ 0	if slope and y-intercept both ≠ 0
Then	then
Text(43, 8, "Tangent line: Y =", round(M, 2), "X", Str2, round(abs(B), 2))	display equation of tangent line
End	end slope ≠ 0 and y-intecept ≠ 0
If M ≠ 0 and B = 0	if slope ≠ 0 and y-intecept = 0

Then	then
Text(43, 8, "Tangent line: Y=", round(M, 2), "X")	display equation of tangent line
End	end slope ≠ 0 and y-intecept = 0
If M = 0 and B ≠ 0	if slope = 0 and y-intecept ≠ 0
Then	then
Text(43, 8, "Tangent line: Y=", Str2, round(abs(B), 2))	display equation of tangent line
End	end slope = 0 and y-intecept ≠ 0
Pause	pause
"" → Y1	clear out Y1

Circle Program – enter a factor of pi, select cos or sin, displays trig circle with the given values for the selected function	Explanation
1 → Q	store 1 in Q, quadrant identifier
Param	turn parametric mode on
"" → X_{1T}	clear X_{1T}
"" → Y_{1T}	clear Y_{1T}
AxesOn	turn axes on
-2.5 →Xmin	initialize Xmin
1.5 → Xmax	initialize Xmax
-1.5 → Ymin	initialize Ymin
1.5 → Ymax	initialize Ymax
ZSquare	use square graphing mode
ClrDraw	clear graph window
0 → Tmin	store 0 to Tmin
2π → Tmax	store 2π to Tmax
.05 → Tstep	initialize Tstep
"cos(T)" → Str1	store "cos(T)" to Str1
Str1 → X_{1T}	store cos(T) to X_{1T}
"sin(T)" → Str1	store "sin(T)" to Str1
Str1 → Y_{1T}	store sin(T) to Y_{1T}
Menu("FUNCTION", "cos", 1, "sin", 2, "End", 0)	set up menu to select cos or sin
Lbl 1	cos part of program
"cos(θ)" → Str1	store "cos(θ)" to Str1
2 → M	store 2 to M
Input "factor of π:", N	input factor of π in fraction or decimal form
fPart(N/2) * 2 → N	convert factor of π to a number between -2π to 2π
If cos(Nπ) < 0	if cos(Nπ) < 0
Then	then
2 → Q	store 2 to Q, quadrant identifier
End	end

cos(Nπ) → I	store cos(Nπ) to I
I → K	store cos((Nπ) to K
√(1 – (cos(Nπ))^2) → J	store the positive sin value corresponding to cos(Nπ) to J
-J → L	store the negative sin value corresponding to cos(Nπ) to L
Goto 3	skip sin part of program
Lbl 2	sin part of program
"sin(θ)" → Str1	store "sin(θ)" to Str1
1 → M	store 1 to M
Input "factor of π:"	input factor of π in fraction or decimal form
fPart(N/2) * 2 → N	convert factor of π to a number between -2π to 2π
If sin(Nπ) < 0	if sin(Nπ) < 0
Then	then
4 → Q	4 → Q, quadrant identifier
Else	else
3 → Q	3 → Q, quadrant identifier
End	end
sin(Nπ) → J	store sin(Nπ) to J
J → L	store sin(Nπ) to L
√(1 – (sin(Nπ))^2) → I	store the positive cos value corresponding to sin(Nπ) to I
-I → K	store the negative cos value corresponding to sin(Nπ) to K
Lbl 3	convert N to a positive number
If N < 0	if N < 0
Then	then
2 – abs(N) → N	convert N to a positive value
End	end
DispGraph	display graph
Line(0, 0, I, J)	draw line between (0, 0) and first (cos(θ), sin(θ))
Line(0, 0, K, L)	draw line between (0, 0) and second (cos(θ), sin(θ))
38 → R	initialize row counter
8 → D	initialize row increment counter
Text(R, 2, "(cos(θ), sin(θ)) =")	display (cos(θ), sin(θ)) =
R + D → R	increment row counter
If Q = 1 or Q = 3	if quadrant 1
Then	then
Text(R, 2, "(", round(I, 3), ",", round(J, 3), ")")	display (I, J)
R + D → R	increment row counter

End	end display (I, J)
If Q = 2 or Q = 3	if quadrant 2
Then	then
Text(R, 2, "(", round(K, 3), ",", round(J, 3), ")")	display (K, J)
R + D → R	increment row counter
End	end display (K, J)
If Q = 2 or Q = 4	if quadrant 3
Then	then
Text(R, 2, "(", round(K, 3), ",", round(L, 3), ")")	display (K, L)
R + D → R	increment row counter
End	end display (K, L)
If Q = 1 or Q = 4	if quadrant 4
Then	then
Text(R, 2, "(", round(I, 3), ",", round(L, 3), ")")	display (I, L)
R + D → R	increment row counter
End	end display (I, L)
If Str1 = "cos(θ)"	if cos
Then	then
Text(10, 2, "θ =", round(N, 3), "π")	display the first value of θ
Text(18, 2, round(2 – N, 3), "π")	display the second value of θ
End	end cos display angles
If Str1 = "sin(θ)"	if sin
Then	then
Text(10, 2, "θ =", round(N, 3), "π")	display the first value of θ
If 1 – N > 0	if 1 – N > 0
Then	then
Text(18, 10, round(1 – N, 3), "π")	display the second value of θ
Else	else
Text(18, 10, round(1 + (2 – N), 3), "π")	display the second value of θ
End	end 1 – N > 0
End	end sin
Text(2, 2, "fn: ", Str1)	display cos or sin
Pause	pause
"" → $X1_T$	clear $X1_T$
"" → Y_{1T}	clear Y_{1T}
Func	turn function mode on
AxesOff	turn axes off
Lbl 0	end program

Step Program – create a stepable unit circle which displays angle, cos, and sin	Explanation
Fix 3	set display to 3 decimal places
Param	set to parametric mode
Degree	set in degree mode
0 → Tmin	store 0 to Tmin
360 → Tmax	store 2π to Tmax
"cos(T)" → Str1	store "cos(T)" to Str1
Str1 → X_{1T}	store cos(T) to X_{1T}
"sin(T)" → Str1	store "sin(T)" to Str1
Str1 → X_{1T}	store sin(T) to Y_{1T}
-2.5 →Xmin	initialize Xmin
1.5 → Xmax	initialize Xmax
-1.5 → Ymin	initialize Ymin
1.5 → Ymax	initialize Ymax
Input "Step in deg:", S	input graph step S in degrees
S → Tstep	store step S to Tstep
Zsquare	use square graphing mode
ClrDraw	clear graph window
AxesOn	turn axes on
DispGraph	display graph
Trace	turn trace on
Pause	pause
"" → $X1_T$	clear $X1_T$
"" → Y_{1T}	clear Y_{1T}
Func	turn function mode on
Radian	turn radian mode on
Float	turn float on
AxesOff	turn axes off