Finger Exercises 24

Plotting

≈ 80 min + stretch

Comp 102 — Programming Fundamentals

What this is. A workout for Lecture 24. You will draw plots by hand, read real plots, match code to pictures, count the piles of a histogram, and find the bugs that stop a plot.

How to do it. On **paper**, **no computer**, in one sitting. If an item takes you more than a minute, skip it and come back. Parts A–S are for *everyone*; Part T is a **stretch**. The Console Quests are for tonight, *at* a computer. The answer key is on the last pages — do not peek until you are done.

Predict first, then check. Write your answer *before* you type anything. A wrong guess teaches you more than an answer you only watched the computer give.

On this sheet. `import matplotlib.pyplot as plt` has already run, and every program ends with `plt.show()` unless an item says something else. In a grid, one small square is 1. Draw a dot as a filled circle, and a line with a ruler or a steady hand.

Part A • Do You Remember?

RECALL ≈ 4 min

A1. The first line of every plotting program: _____

The line that opens the window: _____

A2. Write the **tool** (`scatter`, `plot`, `hist`, `bar`, `pie`):

dots only _____ dots joined by a line _____ piles of **one** list _____

one bar for each group _____ slices of one whole _____

A3. Write the **argument** that does it, with its `=`:

bigger dots _____ triangles, not dots _____ see-through dots _____

twenty piles _____ a bar starts on top of another _____ a name for the legend _____

A4. Write the **function**:

words on top _____ words under the x axis _____ words up the side _____

words at one point inside the plot _____ a line straight up at one x _____

where the x axis starts and ends _____ many small plots in one window _____

A5. Circle **True** or **False**:

(a) `scatter` joins the dots in list order. True / False

(b) Two `scatter` lines before one `show()` draw on the same plot. True / False

(c) `hist` needs two lists, like `scatter`. True / False

(d) `xlim` moves the dots. True / False

(e) A straight line needs only two points. True / False

(f) Two tables with the same average x and average y look the same when plotted. True / False

(g) The engineers showed all 23 launches on the night before. True / False

(h) When the pump's handle came off, the cases were already falling. True / False

Fakhir Shaheen
(sec A, B)

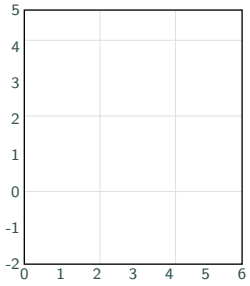
Part B • One Pair, One Dot

DRAW ≈ 5 min

x: how far **across**. y: how far **up**. Item k of `xs` goes with item k of `ys`.

B1.

```
xs = [1, 2, 4, 5]
ys = [3, -1, 4, 2]
plt.scatter(xs, ys)
```

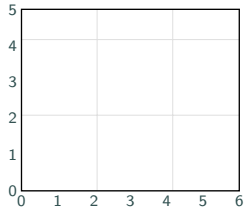


Write the pairs: _____ _____ _____ _____

Draw the dots on the grid. Where is the dot with a **negative** y? _____

B2.

```
xs = [1, 3, 3, 5, 3]
ys = [2, 4, 4, 1, 4]
plt.scatter(xs, ys, alpha=0.5)
```



How many pairs? _____ How many dots can you **see**? _____

Which dot looks darkest? _____ Why?

_____

Draw it. Make the darkest dot big.

B3. Same **dots** as `plt.scatter([1, 2, 4], [3, 5, 2])`? Tick one in each row.

(a) `plt.scatter([4, 1, 2], [2, 3, 5])` ☐ yes ☐ no

(b) `plt.scatter([3, 5, 2], [1, 2, 4])` ☐ yes ☐ no

(c) `plt.scatter([1, 2], [3, 5])` then `plt.scatter([4], [2])` ☐ yes ☐ no

(d) `plt.scatter([1, 2, 4], [3, 5])` ☐ yes ☐ no

In (c), what colour is the dot (4, 2)? _____ In (d), what happens? _____

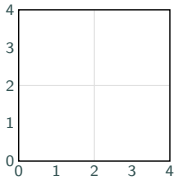
Part C • Join Them in List Order

DRAW ≈ 6 min

`plot` makes the same points as `scatter`, then joins them, **first to second, second to third**, and so on. Draw the line.

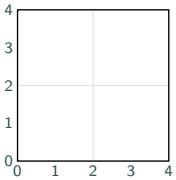
C1.

```
plt.plot([0, 1, 2, 3, 4],
         [0, 3, 1, 3, 0])
```



C2.

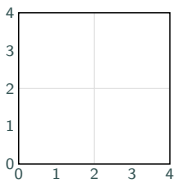
```
plt.plot([0, 4, 0, 4],
         [4, 4, 0, 0])
```



which letter? _____

C3.

```
plt.plot([0, 4, 4, 0, 0],
         [0, 0, 3, 3, 0])
```

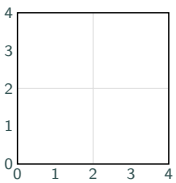


Four corners. Why **five** points?

_____

C4.

```
plt.plot([1, 3, 2],
         [1, 1, 3])
```



Is it a closed triangle?

_____

C5. `plt.plot([2], [3])` shows _____. `plt.scatter([2], [3])` shows _____.

`scatter(xs, ys)` dots | `plot(xs, ys)` a line, in list order | `bar(where, heights)` | `hist(values, bins=n)` piles | `pie(counts)` parts of a whole | `plt.show()` last

C6. Same picture as `plt.plot([0, 2, 4], [0, 4, 0])`, the tent?

(a) `plt.plot([4, 2, 0], [0, 4, 0])` ☐ yes ☐ no

(b) `plt.plot([0, 4, 2], [0, 0, 4])` ☐ yes ☐ no

(c) `plt.scatter([0, 2, 4], [0, 4, 0])` ☐ yes ☐ no

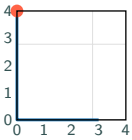
(d) `plt.plot([0, 2], [0, 4])` then `plt.plot([2, 4], [4, 0])` ☐ yes ☐ no

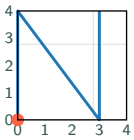
In (d), what colour is each side? _____

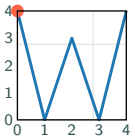
Part D • Write the Lists

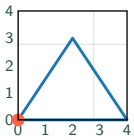
FILL IN ≈ 5 min

Each picture was drawn by **one** `plt.plot` line. Write it. Start at the red end.

D1 

D2 

D3 

D4 

D1. `plt.plot(`_____, _____`)`

D2. `plt.plot(`_____, _____`)`

D3. `plt.plot(`_____, _____`)`

D4. `plt.plot(`_____, _____`)` How many points? ____

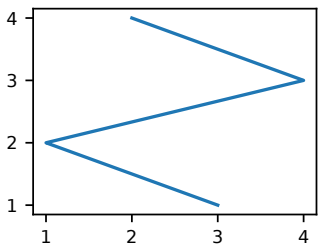
D5. A letter **T**: a top bar from (0, 4) to (4, 4) and a stem from (2, 4) down to (2, 0). Can **one** `plot` line draw it? ☐ yes ☐ no If yes, write it: _____

Part E • Which Code Drew It?

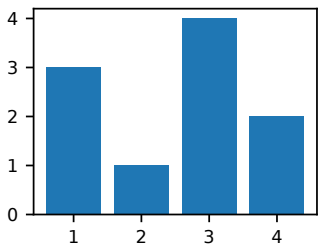
CLASSIFY ≈ 5 min

`a = [1, 2, 3, 4]` and `b = [3, 1, 4, 2]`. Each picture below is one of the eight lines. Write the line's number under each picture. **Two** lines have no picture here. *(Real matplotlib output; only the tick numbers were tidied.)*

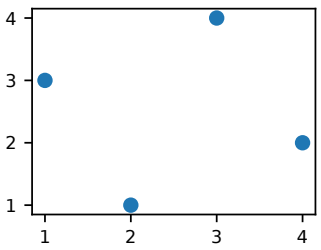
1 `plt.scatter(a, b)` **2** `plt.plot(a, b)` **3** `plt.plot(b, a)` **4** `plt.bar(a, b)`
5 `plt.pie(b)` **6** `plt.hist(b, bins=4)` **7** `plt.scatter(b, a)` **8** `plt.bar(b, a)`



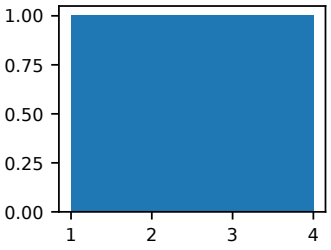
A line ____



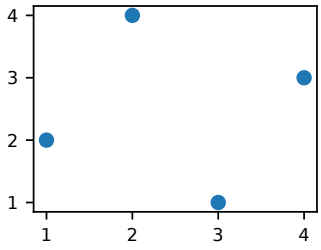
B line ____



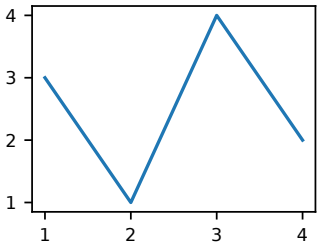
C line ____



D line ____



E line ____



F line ____

E1. Write the number under each picture above.

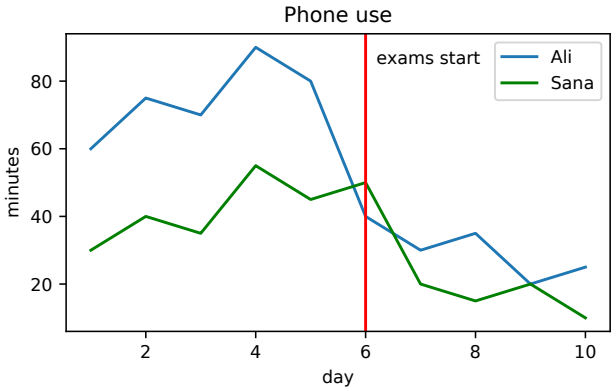
E2. Which two lines have no picture? _____ and _____.

Say what line 8 would draw: _____

`scatter(xs, ys)` dots | `plot(xs, ys)` a line, in list order | `bar(where, heights)` | `hist(values, bins=n)` piles | `pie(counts)` parts of a whole | `plt.show()` last

Part F • Read the Plot

READ ≈ 6 min




```
plt._____(days, ali, label='Ali')
plt.plot(days, sana, color='_____',
         label='_____')
plt.axvline(____, color='red')
plt.text(____, ____, '__________')
plt._____('Phone use')
plt.xlabel('________')
plt.ylabel('________')
plt._____()
```

F1. Fill in the code from the picture. For the words' place, about is enough.

F2. On day 2, who used the phone more? _____ By how many minutes? _____

F3. On which day did Sana use the phone most? _____

F4. Read it: what happened after the exams started? _____

F5. Delete `label='Sana'` only. What does the legend box show now? _____ Delete `plt.legend()` too? _____

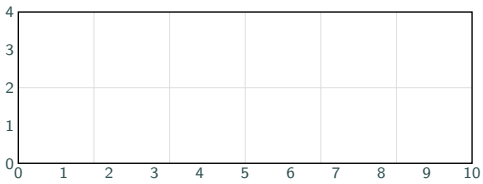
F6. `sana` had only 9 numbers by mistake. What happens? _____

Part G • Where Is It?

DRAW ≈ 5 min

G1.

```
plt.scatter([2, 7, 12], [1, 3, 2])
plt.xlim(0, 10)
plt.axvline(4, color='red')
plt.text(8, 1, 'me')
```



Draw what you see on the box. (The y axis goes from 0 to 4.)

How many dots can you see? ____ How many are in the lists? ____

G2. `plt.xlim(10, 0)` instead. Which number is at the **left** end of the x axis? ____ What happens to the picture? _____

G3. After `plt.xlim(0, 10)`, where can you see `plt.axvline(-1)`? _____
And `plt.text(15, 2, 'hi')`? _____

G4. Challenger: write the three lines that make the x axis go from −10 to 35, draw a red line at launch day (−1), and write `cold` at the point (0, 2).

_____ _____ _____

Part H • Lists from a Loop

TRACE ≈ 7 min

Real lists are built by a loop. Write `xs` and `ys`, then write the shape: **straight**, **curve**, **valley**, **zigzag**, or the **error**. Each program starts with `xs = []` and `ys = []` and ends with `plt.plot(xs, ys)`.

H1.

```
for x in range(5):
    xs.append(x)
    ys.append(x * x)
```

xs _____

ys _____

shape _____

H2.

```
for x in range(-2, 3):
    xs.append(x)
    ys.append(x * x)
```

xs _____

ys _____

shape _____

H3.

```
for k in range(4):
    xs.append(k)
    ys.append(10 - 3 * k)
```

xs _____

ys _____

shape _____

H4.

```
for k in range(6):
    xs.append(k)
    ys.append(k % 2)
```

xs _____

ys _____

shape _____

H5.

```
total = 0
for day in range(1, 6):
    total += day
    xs.append(day)
    ys.append(total)
```

xs _____

ys _____

shape _____

H6.

```
for k in range(1, 5):
    xs.append(k)
ys.append(k * 2)
```

xs _____

ys _____

shape _____

Part I • Only Some Launches

TRACE ≈ 6 min

The engineers' plot, as a program, on six launches:

```
temps  = [21, 12, 24, 18, 27, 14]
damage = [ 1,  2,  0,  0,  0,  1]
some_temps = []
some_damage = []
for i in range(len(temps)):
    if damage[i] > 0:
        some_temps.append(temps[i])
        some_damage.append(damage[i])
plt.scatter(some_temps, some_damage)
```

Each item changes the program above **one** way, and starts again from it. Write the two lists, or the error.

I1. As written: `some_temps` _____ `some_damage` _____

I2. Line 6 is `if damage[i] == 0:` _____ _____

I3. Line 5 is `for i in range(len(temps) - 1):` _____ _____

I4. Line 8 is `some_damage.append(temps[i])` _____ _____
Where do the dots sit? _____

I5. Line 8 moves 4 spaces left, out of the `if` (still in the loop). What happens? _____

I6. Below 15 °C: ___ launches, ___ with damage. 18 °C or warmer: ___ launches, ___ with damage. Which plot shows this: the one of line 9, or one of all 6 launches? _____

Part J • A Line Through Two Points

TRACE ≈ 5 min

Hubble's line: `plt.plot([0, 2], [0, 1000], color='red')`, speed = 500× distance.

J1. On the line, the speed at distance 1 is _____, and at distance 1.5 it is _____.

J2. A galaxy at distance 0.5 moves at 270. Is its dot above or below the line? _____ Andromeda, (0.28, −220)? _____

J3. The line stops at distance 2. Write a line that goes on to distance 3: _____

J4. Same line as `plt.plot([0, 2], [0, 1000])`?

(a) `plot([0,1,2], [0,500,1000])` ☐yes ☐no (b) `plot([2,0], [1000,0])` ☐yes ☐no
(c) `plot([0,2], [1000,0])` ☐yes ☐no (d) `plot([0,1000], [0,2])` ☐yes ☐no

J5. Write the `plot` line for $y = 2x + 1$, from x = 0 to x = 4: _____

Part K • Slices of a Pie

COUNT ≈ 5 min

A slice's share of the circle is its count ÷ the total. A whole circle is 360 degrees.

K1. Nightingale: `plt.pie([14476, 1758, 1748])`. Total _____. Disease is about _____% of the deaths, so its slice is about _____ degrees.

K2. Write the degrees of each slice, or the error.

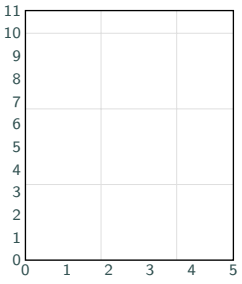
	Line	Slices (degrees), or the error
(a)	`plt.pie([1, 1, 2])`	
(b)	`plt.pie([10, 20, 30])`	
(c)	`plt.pie([1, 2, 3])`	
(d)	`plt.pie([3, 0, 1])`	
(e)	`plt.pie([50])`	
(f)	`plt.pie([2, -1, 3])`	
(g)	`plt.pie([0, 0])`	
(h)	`plt.pie([1, 2, 3], labels=['a', 'b'])`	

K3. Which two pies of K2 look exactly the same? _____ Why? _____

Part L • Bars on Bars

DRAW ≈ 6 min

```
1 months  = [1, 2, 3, 4]
2 disease = [5, 8, 3, 6]
3 wounds  = [1, 2, 4, 0]
4 plt.bar(months, disease)
5 plt.bar(months, wounds, bottom=disease,
6         color='red')
```



L1. Draw the bars on the grid: blue first, then red on top.

L2. The top of each red bar is at ___ ___ ___ ___. Why is there no red in month 4? _____

L3. Delete `bottom=disease`. Now each red bar starts at 0 and is drawn *in front of* the blue one. In which month is the blue bar hidden completely? _____

L4. A third layer, `other = [2, 1, 1, 1]`, needs `bottom=` the top of the red bars. Omar writes `bottom=disease + wounds`. What is `disease + wounds`? _____ What hap-

pens? _____

L5. Fill in the loop that makes `tops = [6, 10, 7, 6]`:

```
tops = []
for i in range(__________):
tops.append(____________________)
```

L6. `plt.bar([1, 2], [3, -2])`: what does the second bar look like? _____

Part M • Piles by Hand

COUNT ≈ 7 min

How `hist` makes its piles. Take the **smallest** and the **largest** value. Cut the space between them into `bins` parts of the **same width**. Count the values in each part. A value that sits **on a cut** goes into the pile on its **right**. The largest value goes into the **last** pile.

M1. `plt.hist([1, 2, 2, 3, 7, 8, 9, 9], bins=4)`
width of a pile ____ cuts: 1, ___, ___, ___, 9 heights: ___ ___ ___ ___

M2. `plt.hist([2, 4, 4, 6, 8, 10], bins=4)` cuts _____ heights _____

M3. The marks of 8 students, `[35, 42, 58, 61, 64, 70, 88, 95]`, with `bins=3`.
cuts _____ heights _____

M4. `plt.hist([0, 10], bins=5)`: heights _____
`plt.hist([6, 1, 4, 9, 3], bins=1)`: heights _____

M5. The heights of all the piles always add up to _____.

Part N • Same Averages, Different Shapes

COUNT ≈ 6 min

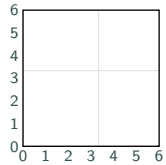
Four small tables, five points each, like the thirteen tables of the lecture.

	points (x, y)					average x	average y
P	(1, 1)	(2, 2)	(3, 3)	(4, 4)	(5, 5)	_____	_____
Q	(3, 1)	(3, 5)	(1, 3)	(5, 3)	(3, 3)	_____	_____
R	(1, 5)	(2, 4)	(3, 3)	(4, 2)	(5, 1)	_____	_____
S	(1, 1)	(5, 1)	(1, 5)	(5, 5)	(3, 3)	_____	_____

N1. Fill in the averages (`sum(xs) / len(xs)`).

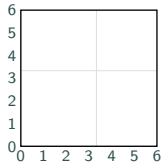
N2. Draw each table as a `scatter`. Name each shape in one word.

P



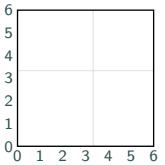
_____

Q



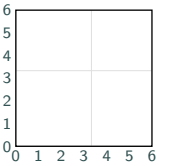
_____

R



_____

S



_____

N3. `plt.hist(xs, bins=5)` for the x values of P, then of Q. Heights for P _____, for Q _____.

N4. Write a table of **three** points with average x 2 and average y 2 that are **not** on one straight line.
_____

Part O • Places in the Window

TRACE ≈ 4 min

O1. `plt.subplot(3, 5, k)` cuts the window into 3 rows and 5 columns. Write each place's `k`.



O2. `plt.subplot(3, 5, 7)` is in row ___, column ___. In `plt.subplot(2, 3, k)`, the bottom-right place is `k` = ___. The lecture put 13 tables in 3 by 5: which places stay empty? _____

O3. The loop of the lecture, changed one way each time. What happens?

(a) `k = 0` at the start, not `k = 1` _____

(b) the line `k += 1` is forgotten _____

(c) `plt.subplot(2, 2, k)` with the 13 tables _____

Part P • Round Plots

TRACE ≈ 3 min

P1. A full turn is _____. With 4 wedges, `step = 6.28 / 4` = _____, and the loop `for k in range(4): angles.append(k * step)` makes `angles` = _____.

P2. With 12 wedges, which `k` points the opposite way to `k = 0`? ____

P3. `disease` has 24 months. `disease[:12]` is the first year. Write the second year: _____

Part Q • Which Tool?

CLASSIFY ≈ 4 min

Write the best tool: `scatter`, `plot`, `hist`, `bar`, `pie`, a **rose** (`bar` on a polar plot) or `subplot`.

	The data	Tool
1	The price of a dollar in rupees on each day of one year	
2	The ages of 300 guests at a wedding: how do they spread?	
3	Hours of sleep and marks of 40 students: do they go together?	
4	How a family's monthly money is split: rent, food, school, other	
5	Goals scored by each of 6 teams	
6	Rain in each of the 12 months, drawn as the year goes round	
7	The places of 200 schools on a map of Lahore (across, up)	
8	Six cities, one small temperature plot for each, side by side	
9	How many people die on each of 24 days of a cholera outbreak, as it rises and falls	

Part R • Error, or What Picture?

FIX ≈ 6 min

Write the **error's name**. If there is no error, say in a few words what goes wrong in the picture. `xs` and `ys` hold 5 numbers each.

	Code	Error, or what you see
1	`import matplotlib as plt`, then `plt.plot(xs, ys)`	
2	`plt.plot([1, 2, 3], [4, 5])`	
3	`plt.scatter([1, 2, 3])`	
4	`plt.Scatter(xs, ys)`	
5	`plt.xlabel(year)` (there is no variable `year`)	
6	`plt.scatter(xs, ys, alpha=2)`	
7	`plt.hist(xs, bins=0)`	
8	`plt.scatter(xs, ys, color='rde')`	
9	`plt.plot(xs, ys)`, then `plt.legend()`	
10	a program with every plot line but no `plt.show()`	
11	`plt.show()`, then `plt.title('Snow')`	
12	`plt.xlabel('day')`, then `plt.xlabel('week')`	
13	`plt.scatter([], [])`	

Part S • One Sentence Each

EXPLAIN ≈ 3 min

S1. Snow's list had the same 578 addresses as his map. Why did the map show the pump when the list did not?

S2. What was wrong with the engineers' plot of 7 launches?

S3. Thirteen tables had the same two averages. Why plot them at all?

Part T • Write the Code

STRETCH ≈ 10 min, optional

Plan each one on paper, then type it in Thonny and run it.

T1. Plot $y = x \times x$ for x from -3 to 3, with a title and both labels. Build the two lists with a loop. Sketch what you expect first.

T2. Roll two dice 1000 times (Lecture 23's `random.randint`), keep each total in a list, and draw `plt.hist(totals, bins=11)`. Guess the shape first.

T3. Count the letters of a word in a dictionary (Lecture 17), then draw one bar per letter: `plt.bar(range(len(letters)), counts)` and `plt.xticks(range(len(letters)), letters)`.

Console Quests

tonight · 5 min · at a computer

When unsure, just try it in the console. These are deliberately **not** answered anywhere on this sheet — the console is the answer key. Install `matplotlib` first (Tools → Manage packages).

1. `plt.plot([3, 1, 4, 1, 5])`, with **one** list. What does Python use for x?
2. `plt.plot(xs, ys, marker='o')`: what changes?
3. Try `plt.grid()` before `plt.show()`.
4. `plt.savefig('snow.png')` before `plt.show()`. Find the file next to your program and open it.
5. Roll one die 6000 times and draw `plt.hist(rolls, bins=6)`, then `bins=5`. Why does `bins=5` look wrong?

Answer Key

Check your work *after* attempting everything. If an answer surprises you, that item is worth re-reading in the slides. Console Quests have no key — the console is the key.

A1. `import matplotlib.pyplot as plt`; `plt.show()`.

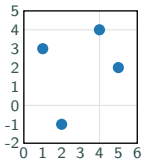
A2. `scatter`; `plot`; `hist`; `bar`; `pie`.

A3. `s=80` (any bigger number); `marker='^'`; `alpha=0.5`; `bins=20`; `bottom=`; `label=`.

A4. `title`; `xlabel`; `ylabel`; `text`; `axvline`; `xlim`; `subplot`.

A5. (a) **False** — that is `plot`. (b) **True** (Snow's map: deaths, then pumps). (c) **False** — one list; it counts. (d) **False** — the dots stay; we see more or less of the axis. (e) **True** (Hubble: (0, 0) and (2, 1000)). (f) **False** — table 13 was a dinosaur. (g) **False** — only the 7 with damage. (h) **True** — read a plot honestly.

B1. (1, 3), (2, −1), (4, 4), (5, 2). The dot (2, −1) is **below the 0 line**.



B2. **5** pairs, but only **3** dots: (1, 2), (3, 4), (5, 1). **(3, 4)** is darkest: three see-through dots sit on top of each other (the Challenger "x2").

B3. (a) **yes**: the same three pairs in another order; `scatter` does not care about order. (b) **no**: x and y swapped, so (3, 1), (5, 2), (2, 4). (c) **yes**, the same places, but the dot (4, 2) is **orange**: each new `scatter` takes the next colour. (d) **no**: 3 numbers and 2 numbers, a **ValueError** ("x and y must be the same size").

C1. An **M**: up to (1, 3), down to (2, 1), up to (3, 3), down to (4, 0).



C2. A **Z**: across the top, back down the diagonal to (0, 0), across the bottom. The x values go back to 0, so the line goes back to the left.



C3. A closed **rectangle**, 4 wide and 3 high. The fifth point, (0, 0) again, draws the last side back to the start.

C4. **Not closed**: two sides only, (1, 1) to (3, 1) to (2, 3). The side from (2, 3) back to (1, 1) is missing: add the first point again at the end.

C5. **Nothing** (an empty plot, no error): a line needs two points. **One dot** at (2, 3).

C6. (a) **yes**: the same tent, drawn from right to left. (b) **no**: (0, 0) to (4, 0) to (2, 4), a floor and the right side only. (c) **no**: three dots, no line. (d) **yes** in shape, but in two colours: the left side **blue**, the right side **orange**.

D1. `plt.plot([0, 0, 3], [4, 0, 0])`: an L.

D2. `plt.plot([0, 0, 3, 3], [0, 4, 0, 4])`: an N.

D3. `plt.plot([0, 1, 2, 3, 4], [4, 0, 3, 0, 4])`: a W.

D4. `plt.plot([0, 4, 2, 0], [0, 0, 3, 0])`: **4** points for 3 corners; the first point comes again to close it.

D5. **Yes**: `plt.plot([0, 4, 2, 2], [4, 4, 4, 0])`. The line goes back over the top bar from (4, 4) to (2, 4), then down. A line may run over itself.

E1. A **3** (x is `b`: the line zigzags left and right as it climbs); B **4**; C **1**; D **6** (each value once, so four piles of 1); E **7** (x and y swapped); F **2**.

E2. **5** and **8**. (Line 5 is a pie of four slices, 3, 1, 4 and 2 parts of 10.) Line 8: bars at x = 3, 1, 4, 2 with heights 1, 2, 3, 4, so from left to right the heights are 2, 4, 1, 3.

F1. `plot`; `'green'`, `'Sana'`; `6`; about `6, 85` (it was `6.2, 85`), `'exams start'`; `title`; `'day'`; `'minutes'`; `legend`.

F2. **Ali**: 75 against 40, **35** minutes more.

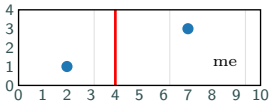
F3. Day **4** (55 minutes).

F4. Both used the phone **less**: both lines fall after day 6.

F5. Only **Ali** (a line with no label is left out of the legend). Without `legend()`: **no box** at all.

F6. A **ValueError** at Sana's `plot` line: `days` has 10 numbers, `sana` 9.

G1. Dots at (2, 1) and (7, 3); a red line straight up at x = 4; the word `me` starting at (8, 1). **2** dots seen, **3** in the lists: (12, 2) is right of the axis's end.



G2. **10**. The axis runs backwards, so the picture is **flipped** left to right.

G3. **Nowhere** for both, and no error: −1 and 15 are outside 0 to 10. `text` uses the plot's numbers, not centimetres.

G4. `plt.xlim(-10, 35)`; `plt.axvline(-1, color='red')`; `plt.text(0, 2, 'cold')`.

H. H1 `[0, 1, 2, 3, 4]`, `[0, 1, 4, 9, 16]`: a **curve** that gets steeper. H2 `[-2, -1, 0, 1, 2]`, `[4, 1, 0, 1, 4]`: a **valley** (a U), lowest at x = 0. H3 `[0, 1, 2, 3]`, `[10, 7, 4, 1]`: **straight**, going down 3 at each step. H4 `[0, 1, 2, 3, 4, 5]`, `[0, 1, 0, 1, 0, 1]`: a **zigzag**. H5 `[1, 2, 3, 4, 5]`, `[1, 3, 6, 10, 15]`: a **curve** up; each step is bigger than the last. H6 `[1, 2, 3, 4]`, `[8]`: the last line is *outside* the loop, so it runs once, with k = 4. 4 numbers and 1 number: **ValueError**.

I1. `[21, 12, 14]` and `[1, 2, 1]`: 3 dots.

I2. `[24, 18, 27]` and `[0, 0, 0]`: 3 dots on the bottom line, all warm.

I3. `[21, 12]` and `[1, 2]`: the last launch, (14, 1), is never looked at.

I4. `[21, 12, 14]` and `[21, 12, 14]`: dots on the **diagonal** (x = y), far above 2. No error, just a wrong picture.

I5. `some_damage` gets all 6 numbers, `some_temps` only 3: **ValueError** at `scatter`. (Moving line 7 instead

gives an **IndentationError**: the `if` would have no body.)

I6. Below 15: **2** launches, **2** with damage. 18 or warmer: **4**, **1** with damage. Only the plot of **all 6** shows it: line 9's plot has no clean launch at all.

J1. **500**; **750**.

J2. **Above**: the line is at 250 there. Andromeda is **below the 0 line**: a speed under 0 means it comes closer.

J3. `plt.plot([0, 3], [0, 1500], color='red')`.

J4. (a) **yes**: the middle point is on the line too. (b) **yes**: drawn from the other end. (c) **no**: it *falls* from (0, 1000) to (2, 0). (d) **no**: x and y swapped, an almost flat line that goes far to the right.

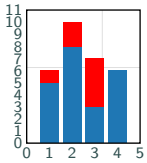
J5. `plt.plot([0, 4], [1, 9])`. Two points are enough; more points on the same line are fine too.

K1. **17,982**; about **80**% ($14476 \div 17982 = 0.805$); about **290** degrees.

K2. (a) **90, 90, 180**. (b) **60, 120, 180**. (c) **60, 120, 180**: the same pie as (b). (d) **270, 0, 90**: the 0 gets no slice (its label is still written). (e) **360**: one whole circle. (f) **ValueError**: no negative slices. (g) **ValueError**: all slices are 0, nothing to share. (h) **ValueError**: 3 counts but 2 labels.

K3. **(b) and (c)**. A pie shows only the **shares**: 10, 20, 30 and 1, 2, 3 are both 1/6, 2/6, 3/6. A pie hides how big the total is.

L1.



L2. **6, 10, 7, 6**. Month 4 has 0 wounds: a red bar of height **0**.

L3. Month **3**: the red bar (4) is taller than the blue one (3). In months 1 and 2 the blue sticks out above the red; in month 4 there is no red.

L4. `[5, 8, 3, 6, 1, 2, 4, 0]`: `+` on two lists **joins** them, it does not add the numbers. 4 bars, 8 bottoms: a **ValueError**.

L5. `range(len(months))` (or `range(4)`); `tops.append(disease[i] + wounds[i])`. Then `bottom=tops` for `other`.

L6. It goes **down** from 0 to −2. No error.

M1. Width $(9 - 1) \div 4 = 2$; cuts 1, **3, 5, 7**, 9; heights **3, 1, 0, 4** (the 3 sits on a cut and goes right; a pile can be empty).

M2. Cuts 2, 4, 6, 8, 10; heights **1, 2, 1, 2**. Both 4s go right of the cut at 4; the 10 stays in the last pile.

M3. Width $60 \div 3 = 20$; cuts 35, 55, 75, 95; heights **2, 4, 2**.

M4. **1, 0, 0, 0, 1**: two lonely piles at the ends. **5**: one pile holds everything.

M5. The **number of values** in the list (8 in M1, 6 in M2, 8 in M3). Every value lands in exactly one pile.

N1. **3.0** and **3.0** for all four tables.

N2. P a line **going up**; Q a **plus** sign; R a line **going down**; S an **X** of corners (the five on a die). Four shapes, one pair of averages.

N3. P `[1, 2, 3, 4, 5]`: **1, 1, 1, 1, 1**, flat. Q `[3, 3, 1, 5, 3]`: **1, 0, 3, 0, 1**, a pile in the middle. Same average 3, not the same list.

N4. Many answers, e.g. (1, 1), (3, 1), (2, 4): $(1 + 3 + 2) \div 3 = 2$ and $(1 + 1 + 4) \div 3 = 2$. Check yours by adding.

O1. Row by row, left to right: 1–5 on top, 6–10 in the middle, 11–15 at the bottom.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

O2. Row **2**, column **2**. **6**. Places **14 and 15**.

O3. (a) **ValueError** at the very first subplot: there is no place 0 (places are 1 to 15). (b) No error: all 13 tables are drawn in **place 1**, on top of each other. (c) Four plots, then a **ValueError** at the 5th table: 2 by 2 has only places 1 to 4.

P1. **6.28**; **1.57**; `[0.0, 1.57, 3.14, 4.71]`. The angle 6.28 is never used: it is the same place as 0.

P2. `k = 6`: $6 \times 0.523 = 3.14$, half a turn.

P3. `disease[12:]`.

Q. 1 `plot` (how it changes). 2 `hist` (one list, how it spreads). 3 `scatter`. 4 `pie` (parts of one whole; `bar` is fine too). 5 `bar`. 6 a **rose**: data that goes round. 7 `scatter` (Snow's map). 8 `subplot`. 9 `plot` (Snow's days; `bar` is fine too).

R. 1 **AttributeError**: the plot tools are in `matplotlib.pyplot`, not `matplotlib`. 2 **ValueError** (3 and 2 numbers). 3 **TypeError**: `scatter` needs `ys` too. 4 **AttributeError**: Python cares about capitals. 5 **NameError**: the word needs quotes, `'year'`. 6 **ValueError**: `alpha` goes from 0 to 1. 7 **ValueError**: at least one pile. 8 **ValueError**: no colour is called `rde`. 9 No error, but **no legend box**: no line has a `label=` (Python only warns). 10 No error, but **no window** opens. 11 The window opens with **no title**: the title comes after the picture was shown. 12 Only **week** is shown: the second call replaces the first. 13 No error: an **empty** plot.

S1. On the map the eye sees *where* the dots crowd, all at once; in a list you only read one address at a time.

S2. It left out the 16 launches with no damage, and all of those were warm; without them, cold did not look dangerous.

S3. An average is one number; it hides the shape. Only the plots showed lines, a star, a circle and a dinosaur.

T1. A loop `for x in range(-3, 4):` appends `x` and `x * x`; then `plt.plot(xs, ys)`, `title`, `xlabel`, `ylabel`, `show`. A valley: (0, 0) at the bottom, 9 at both ends.

T2. A **hill**, tallest at 7 (6 of the 36 pairs make 7), lowest at 2 and 12 (1 pair each). 11 bins: one pile for each total from 2 to 12.

T3. `letters = list(d.keys())` and `counts = list(d.values())` after the counting loop. For `'banana'`: bars of 1, 3, 2 for b, a, n.