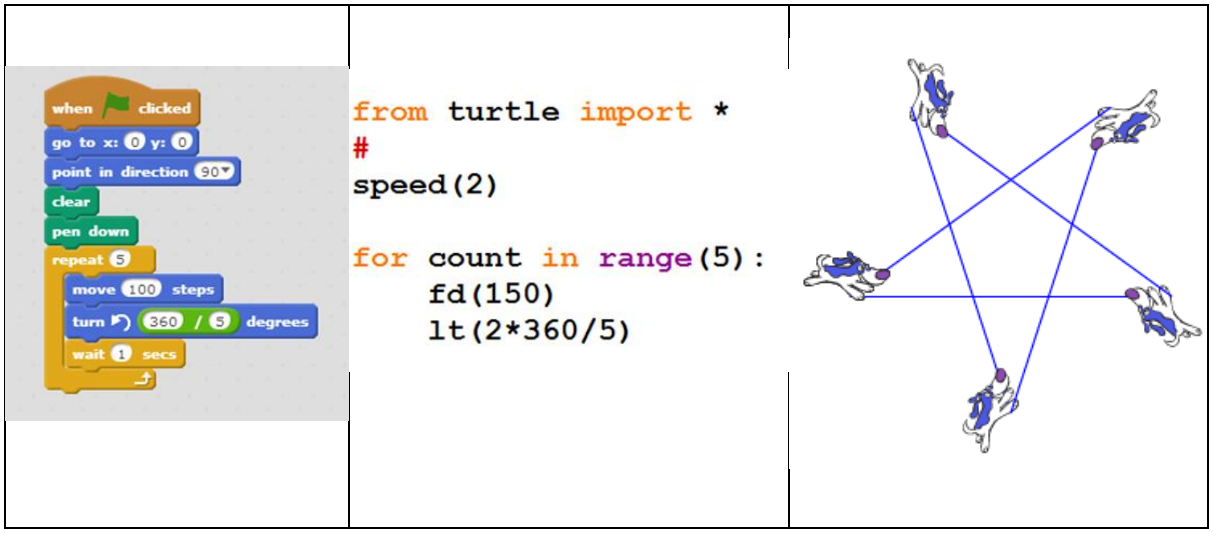
COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
In collaboration with BCS, The Chartered Institute for IT

UCL ENGINEERING
Change the world

UCL/CAS Training for Teachers
Algorithms and Programming Module 1



WORKBOOK 12
REACH FOR THE STARS

- Addressed to Teachers
- Activities are graded: easy to hard – 0 to 5*. You should attempt every activity marked without a star or marked with one *.

ispython.com
CC BY NC SA

COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
In collaboration with BCS, The Chartered Institute for IT

UCL ENGINEERING
Change the world

UCL/CAS Training for Master Teachers
Algorithms and Programming Module 1

CONTENTS

CC BY NC SA

MISSION 1: SHORTHAND PROGRAM FOR A STAR

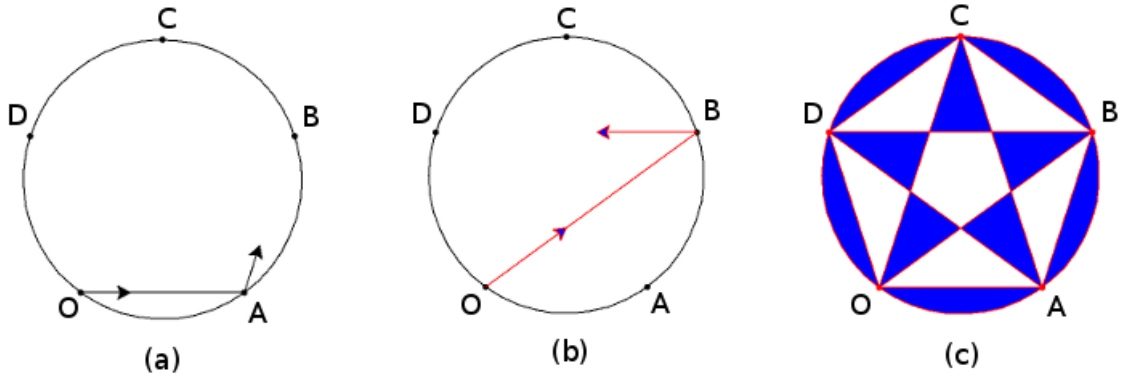


Figure 1. ***action geometry:*** **for the pentagon and pentagram**

CRACK THE CODE

- Figure 1 is taken from Workbook 9: Action Geometry: Joining the Dots – Computational Thinking. By mimicking the sprite/turtle drawing unplugged, we attempted to evaluate the turning angles and so classify polygons and polygrams (stars).

- We summarise the argument to find the sprite/turtle turning angle for the pentagon in Figure 1(a):
  1. In tracing out the pentagon OABCDO, and finally turning to face along OA again, the sprite/turtle has turned through 360 degrees – SPIN(360).
  2. Run the Scratch program below in Figure 2(a) to illustrate the SPIN(360)
  3. During this tracing, our sprite/turtle has turned through the same angle at each of the 5 vertices A, B, C, D, O.
  4. Consequently, the turning angle for the pentagon at each vertex is 360/5.
  5. Try cracking the code in Figures (2) and (3).

ispython.com
CC BY NC SA

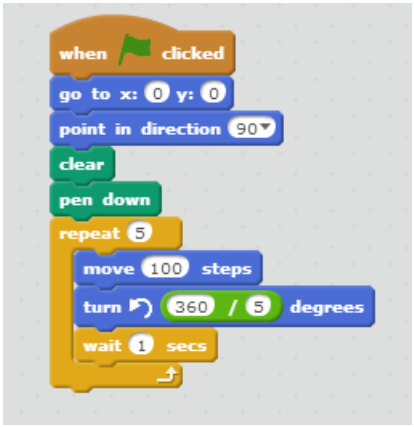


Figure 2(a). scratch 2.0 program for a pentagon

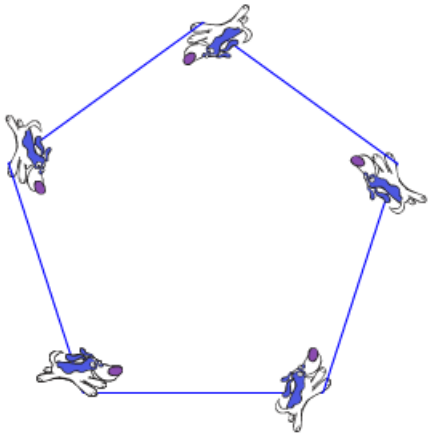


Figure 2(b). a sprite pentagon

- Notes for Figure 2:
  - (a) Plug in to Scratch 2.0 is available in Workbook 6.
  - (b) The first 4 instructions are included to make sure the program can rerun with a clear screen and with the sprite positioned at O (0, 0) and facing to the right
  - (c) The wait 1 second instruction is included in the loop to slow the program down at each step so that you can see the sprite make its turns.
  - (d) The instructions in the repeat loop are executed 5 times.
  - (e) The angle of turn is expressed in the form 360/5 to emphasise both how the angle is calculated and to pick out the symmetry in loop as in the shorthand form (5, 360/5). We call this the 'pentagon' loop. The format also points the way for generalising for polygons and polygrams (stars).
  - (f) In order to insert 360/5 in the loop of the Scratch program, you will need to go to the operator category to extract the divisor operation, and insert 360 and 5 into it. See Workbook 13.

```
from turtle import *
#
speed(2)

for count in range(5):
    fd(100)
    lt(360/5)
```

Figure 3(a). python program for a pentagon

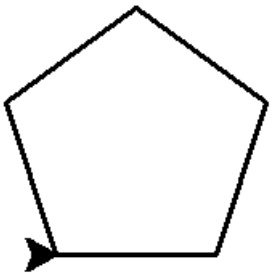


Figure 3(b). a turtle pentagon

ispython.com
CC BY NC SA

- Arguing from the symmetry of the pentagram (5-star) in Figure 1(c), and along similar lines to the way we find the sprite/turtle turning angle for the simple regular pentagon, ***(generalising),*** we trace the pentagram OBDACO in Figure 1(b). It is similarly evident that in drawing the pentagram (5-star) that: we turn through 5 equal angles at B, D, A, C, O. (if we include turning to face B when we return to O).
  It's not so obvious how much the sprite/turtle spins in total in turning through these 5 equal angles. One way is to count how many times we go round the centre of the star. Or if you look at Figure 1(c) that we go two laps (twice around) the small central pentagon. And that corresponds to joining every second point. Or in Figure 4(c), we can see how Terri turns left through an angle at each vertex and by the time she reaches the third vertex has turned round completely once and twice by the final turn at the fifth vertex.
  But we are now able to get the sprite/turtle to help by making our action geometry come alive. To see the turning dynamically, but slowly, run the Scratch program in Figure 4(a):

Figure 4(a). scratch 2.0 program for a pentagram

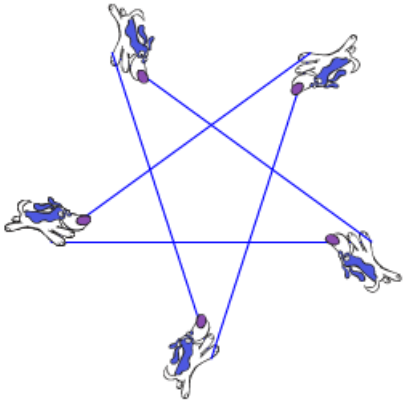


Figure 4(b). a sprite pentagram (5-star)

(g) In order to insert 2*360/5 in the loop of the Scratch program, you will need to go to the operator category to extract the multiplier operation, and insert 2 in the first place and then 360/5 in the second place. Reference Workbook 6.

ispython.com
CC BY NC SA

```
from turtle import *
#
speed(2)

for count in range(5):
    fd(150)
    lt(2*360/5)
```

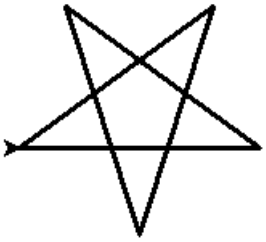


Figure 5(a). a python program for a pentagram

Figure 5(b). a turtle pentagram

- So if we SPIN twice round, that's 2*360 degrees -- SPIN(2*360).
  And if we turn 5 equal times at the vertices of pentagram to achieve that total spin of 2*360 degrees, the turning angle at each vertex is the total spin divided by 5, i.e. 2*360/5.
  And our shorthand program for the pentagram is

  `(5, fd(150), 2*360/5)`.....`program(4s)`

  We call this repeat loop the 'pentagram' loop.
  The following Mission can be omitted in a first reading of this Course to learn programming. The Activities are classified as ** to *****. Being able to program in Scratch or Python is an invaluable tool in our attempts to investigate and classify regular stars. Some elementary knowledge of trigonometry will also be needed for the more difficult activities.

ispython.com
CC BY NC SA

MISSION 2: REACHING FOR THE STARS

- **What do we get when we draw program(5s)? How does it relate to the star of program(4s)? How many different pentagrams (5-stars) are there?

```
(5, fd(150), 3*360/5).....program(5s)
```

- ***Crack the code of the following shorthand programs (6s) and (7s) --- (6s) should be straightforward – you may have to run (7s) to work out what it does:

```
(5, fd(100), 1*360/5).....program(6s)

(5, fd(100), 4*360/5).....program(7s)
```

- **Run the two programs (6s) and (7s) in Scratch/Python, and compare the two figures.
- ** What is the relationship in angles between turning left and right in a loop?
- ***What do the last 3 exercises tell you about the different turning angles for a pentagon and pentagram? Does it help in the classification of regular 5-stars?
- ***** Can you generalise your findings to a regular 7-star?

MISSION 3: STARS THICK AND THIN

1. *Crack the Code for program(8s). Write the equivalent Scratch/Python program for program(9s). Run it slowly so that you can see the total SPIN that the sprite/turtle turns through.

```
(7, fd(150), 2*360/7.....program(8s)
```

2. ***Experiment*** with the Scratch program you have written for program(8s), by changing the repeat loop in your program to represent the shorthand program(9s). Run it slowly so that you can see the total SPIN that the sprite/turtle turns through. Is there a difference in the drawing produced by program(8s) and (9s)?

```
(7, fd(150), 3*360/7.....program(9s)
```

3. ***In the heptagram (7-pointed star) we were able to join up every second point as with the pentagram, and we then joined every third point. Using your repeat loop, ***experiment*** and investigate the different stars with 7 points `(7, fd(150) 4*360/7)`, `(7, fd(150), 5*360/7)` `(7, fd(150), 6*360/7)`… How many essentially different ones are there? How many are COMPLETE stars? Fill in Table 2. You may have to alter `fd(150)` to `fd(100)` to make sure the drawing stays on the screen.

ispython.com
CC BY NC SA

****MISSION 4: A STARBURST

1. Try amending the program in Figure 6 for the hexagram (6, fd(150) 2*360/6). Why doesn't it work? How do you make the star of David (hexagram)? The star of David is an INCOMPLETE or COMPOSITE hexagram. In fact, all hexagrams (6-stars) are COMPOSITE. The simple regular hexagon is the only COMPLETE 6-point figure.

*** MISSION 5: CLASSIFY THE STARS

Some interesting classifications:

(a) Simple regular polygons are COMPLETE and can be drawn for any number of (n) points ***(generalisation).***

(b) The pentagram (5-star) is the first COMPLETE star. And there is only one.

(c) The equilateral triangle and the square both are simple regular polygons but only generate incomplete stars

(d) The hexagon is the only simple regular polygon with 5 or more points which produces no COMPLETE stars.

Table 2

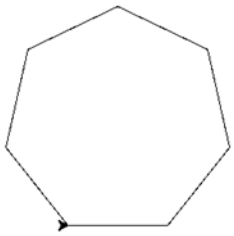
Turning Angle	Repeat value	Regular Shape Star Polygram	Shorthand	Length of side
2*360/5	5	pentagram(5pt)	(5, 2*360/5)	150
?	7	7 point	(7, 2*360/7)	150
?		7 point	(7, 3*360/7)	150
?		7 point	… ?	150

(e)

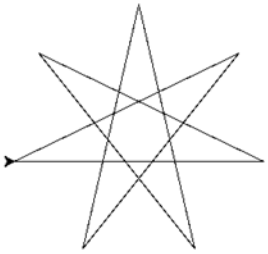
1. *****Classify all the different stars up to 12-point stars into different COMPLETE stars and COMPOSITE STARS.
2. *****Simple regular polygons with a prime number of points produce COMPLETE stars. But how many? And how many different ones? Use the 11-sided star as a starting point to achieve your classification.

ispython.com
CC BY NC SA

```
from turtle import *
#
def heptagon():
    for i in range(7):
        fd(100)
        lt(360/7)

heptagon()

exitonclick()
```



```
from turtle import *
#
def heptagram():
    for i in range(7):
        fd(200)
        lt(3*360/7)

heptagram()

exitonclick()
```



(a) ***heptagon definition and call*** ***(b)'spikiest' thin heptagram definition and call***

Figure 6. Python function definition for heptagon and heptagram

Summary of our unplugged activities in drawing polygons and stars. We experimented with paths where the SPIN(2*360) was used; it corresponds to joining every second point in drawing the pentagram (5-pointed) star, for example, and results in the pet/robot turning a complete circle twice (2*360) in tracing the pentagram. This was ***generalised*** to 3*360, 4*360 or more, for stars with more vertices. The shorthand form for the program to draw a pentagram is

(5, fd(150) 2*360/5).....program(14s)

** MISSION 6: 'SPIKIEST' 9-STAR

i. *Using the format in program (14s) define a function 'pentagram' in Scratch to draw a pentagram.
ii. *** define a function: 'ninestar' (9-sided (pointed) star) (9, fd(200), 2*360/9) and run a program to draw it.
iii. **** define a function: 'spikeyninestar' the spikiest ninestar and run a program to draw it.

CC BY NC SA