Turtle Flowers

Student assessment

Grade: 5 - 8

Language: Blockly or Python turtle.

Assessed concepts:
This coding task assumes knowledge of the below concepts. These are covered in our Intro to Programming in Python or Blockly courses, and in our free short activities:

- Programming the turtle using Blockly or Python to move forwards and/or backwards and turn different angles;
- Changing the pen colour, fill colour, background colour, and/or pen size;
- Using loops to avoid unnecessary repetition of code;
- Using variables to store and reuse information;
- Collecting and using user input.

Optional concepts:

- Differentiating and converting different types of data (integers and strings);
- Performing mathematical calculations with code;
- Using conditionals (using if, else, elif);
- Using nested loops.

Curriculum links:

Australian Curriculum: Digital Technologies - Years 5 and 6:

- Acquire, store and validate different types of data, and use a range of software to interpret and visualise data to create information (ACTDIP016)
- Define problems in terms of data and functional requirements drawing on previously solved problems (ACTDIP017)
- Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)(ACTDIP019)
- Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input (ACTDIP020)
- Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols (ACTDIP022)

Australian Curriculum: Digital Technologies - Years 7 and 8:

- Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030)

Cross-curricular links: Mathematics — shape and geometric reasoning

Project Task: Build your own Flower Bot!

Use your programming skills to build a flower-drawing bot. Your program should use the turtle to draw a flower of your own design, using lines and/or geometric shapes.

You need to:

- Write a program to draw an attractive flower of your own design. Think about using colour, fills, geometry and symmetry when creating your design.
- Collect user input and let the user control some aspects of how the flower looks. For example, the user might be able to choose the number of petals, or the colour of the flower.

There are two parts to this assignment:

Part 1: Planning

- Sketch some early designs for your flower. What angles will the turtle need to turn to draw your design? Include the angle calculations in your sketches.
- Your program should give the user options to decide on how the flower will look. What information will your program collect from the user, and how will your program use this information to change the appearance of the flower? (1–2 paragraphs)

Part 2: Write your program

Write your program using the Python Turtle or Blockly Turtle Playground. Here are some things to consider:

- How well does your program run? Does it have any bugs?
- Is your code clear, logical, and easy for someone else to follow and understand?
- How creative and attractive is your flower design? Have you used colour and symmetry? (Your program may need to perform some calculations!)
- How many options do you give the user to control the design of the flower and how well does your program use this data?
- Does your program avoid unnecessary repetition by using loops where possible?

Grading

Category	*Below Satisfactory*	*Satisfactory*	*Above Satisfactory*
Planning and design	The student has given little thought to the design or to how users will interact with the program. The student has not demonstrated an understanding of symmetry and geometry, and angle calculations are missing or inaccurate. The student has provided minimal description of data to be collected from the user and how this data will be used by the program.	The student has created an attractive design, has demonstrated an understanding of symmetry and geometry, and has accurately calculated the angles needed for the design. The student has considered what data they will need to collect from users, and has outlined how they will use that data.	The student has created a complex and attractive design, has demonstrated a strong understanding of symmetry and geometry, and has accurately calculated the necessary angles. The student has provided a detailed and considered description of how the user will interact with the program, including what data will be collected and how this data will be used.
Algorithm design	The student has not developed an algorithm which satisfactorily addresses the defined problem.	The student has developed an algorithm which satisfactorily addresses the defined problem and that incorporates at least one example of user input and repetition (iteration).	The student has developed an algorithm which comprehensively addresses the defined problem, that incorporates user input and repetition (iteration), and that uses sophisticated programming constructs and concepts beyond those required (e.g. nested loops).
Code execution	The program does not run and/or has several issues or bugs. The program lacks organisation and logic.	The program runs without error and is mostly well organised and logical. The program may have some bugs or minor issues, such as redundancy (unnecessary repetition of code).	The program runs without error and is extremely well-organised and logical.
User interaction	The program does not ask for user input, or user input is ignored. The student does not demonstrate understanding of variables to store information.	The program allows user interaction by asking for user input, using variables to store data, and using the data collected from the user to determine the program's output. The student has provided clear instructions for the user.	The program provides multiple opportunities for user interaction by asking for multiple user inputs, using variables to store data, and using the data collected from the user to determine the program's output. The user instructions are clear and well-written.

Samples

We've created a few sample programs you can try out and tinker with yourself, or show to your students for inspiration.

The first is a simple example which asks the user to choose the colour of the flower, calculates the angle the turtle will need to turn in between drawing each petal, and uses a loop to draw a symmetrical five-petaled flower.

The second, more complex example collects multiple inputs from the user (the number of outer petals, the number of inner petals, and the petal colours), converts strings to integers, calculates angles, and uses nested loops to draw the correct number of inner and outer petals.

program.py

```
from turtle import *

colour = input('What colour is your flower? ')
fillcolor(colour)
begin_fill()
for i in range(5):
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  left(360/5)
end_fill()
```

Submissions | Output | Animation

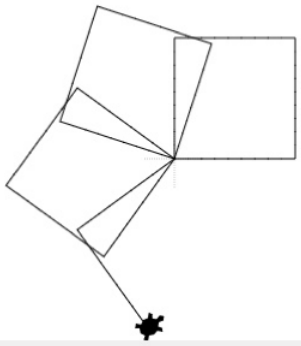
```
What colour is your flower? blue
```

program.py

```
from turtle import *

colour = input('What colour is your flower? ')
fillcolor(colour)
begin_fill()
for i in range(5):
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  left(360/5)
end_fill()
```

Submissions | Output | Animation

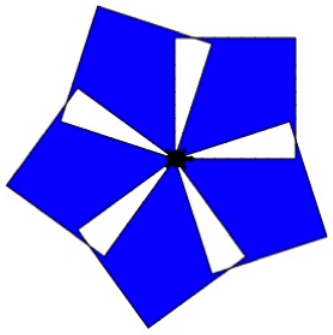


program.py

```
from turtle import *

colour = input('What colour is your flower? ')
fillcolor(colour)
begin_fill()
for i in range(5):
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  forward(100)
  left(90)
  left(360/5)
end_fill()
```

Submissions | Output | Animation



program.py

```
from turtle import *

outpetals = int(input('Number of outer petals: '))
inpetals = int(input('Number of inner petals: '))
outcolour = input('Colour of outer petals: ')
incolour = input('Colour of inner petals: ')
inpetals_angle = 360 / inpetals
outpetals_angle = 360 / outpetals
pensize(4)
pencolor(outcolour)
fillcolour(outcolour)
for i in range(int(outpetals)):
  begin_fill()
  for i in range(2):
    forward(60)
    left(45)
    forward(60)
    left(135)
  end_fill()
  left(outpetals_angle)
pencolor(incolour)
fillcolour(incolour)
for i in range(int(inpetals)):
  begin_fill()
  for i in range(4):
    forward(50)
    left(90)
  end_fill()
  left(inpetals_angle)
```

Submissions | Output | Animation

```
Number of outer petals: 10
Number of inner petals: 8
Colour of outer petals: darkgreen
Colour of inner petals: pink
```