

## **Python Turtle Art Test**

(T) Are photo(s) and commentary of functions used provided?

Demonstrates principle of modularity to design reusable code in computer programs . (Ministry expectation B2.3)

(T) Are photo(s) and commentary of variable(s) used provided?

Demonstrates judgement in using appropriate types of variables (Ministry expectations A1.1)

(T) Are photo(s) and commentary explaining your use of repetition (loop) blocks (and nested blocks) provided?

Demonstrates judgement selecting and using repetition constructs in Turtle Art program. (Ministry expectation A2.2, A2.3)

(T) Have I imported and used existing sub-programs created by others correctly (e.g., random)?

Demonstrate the ability to use existing sub-programs (e.g., random number generator) within programs (Ministry expectation A3.1)

(C) Is my write-up easy to understand? Grammatically correct? Am I using the correct vocabulary?

Demonstrates effective communication through word choice and sentence structure (Ministry expectation B4.6)

(C) Are annotated screenshots provided that explain how different parts of program work together?

Demonstrates effective communication explaining the flow of program. (Ministry expectation B2.4)

(C) Are variables and functions descriptively named with good inline comments explaining them?

Demonstrates effective communication using naming conventions and comments. (Ministry expectation A4.2)

(A) Are photo(s) and commentary of proof of finished drawing provided?

Demonstrates ability to design simple algorithms (Ministry expectation B3.1)

(A) Demonstrates effectiveness interacting with the VI editor and Python environment. (Ministry expectation C3.1)

Are photo(s) and commentary of how VI editor and Python works (editing, executing and saving source) provided?

(A) Did I get all my screen shots and commentary moved into my journal on time?

Demonstrates effective collecting, transferring and embedding information in the Journal (Ministry expectation C2.1)

## About the vi editor and Python

The vi editor is a text editor created for Unix. The vi editor was originally released in 1976. The vi editor stands for visual editor. The vi editor is very interesting because it does not use a mouse, and can be operated entirely by keyboard.

vi commands:

“i” - Enter insert mode, allows you to insert text.

“esc” - Enter command mode, allows you to save documents and complete administrative tasks.

“:w” - The write (save) without quitting.

“:q” - Quit without writing (saving).

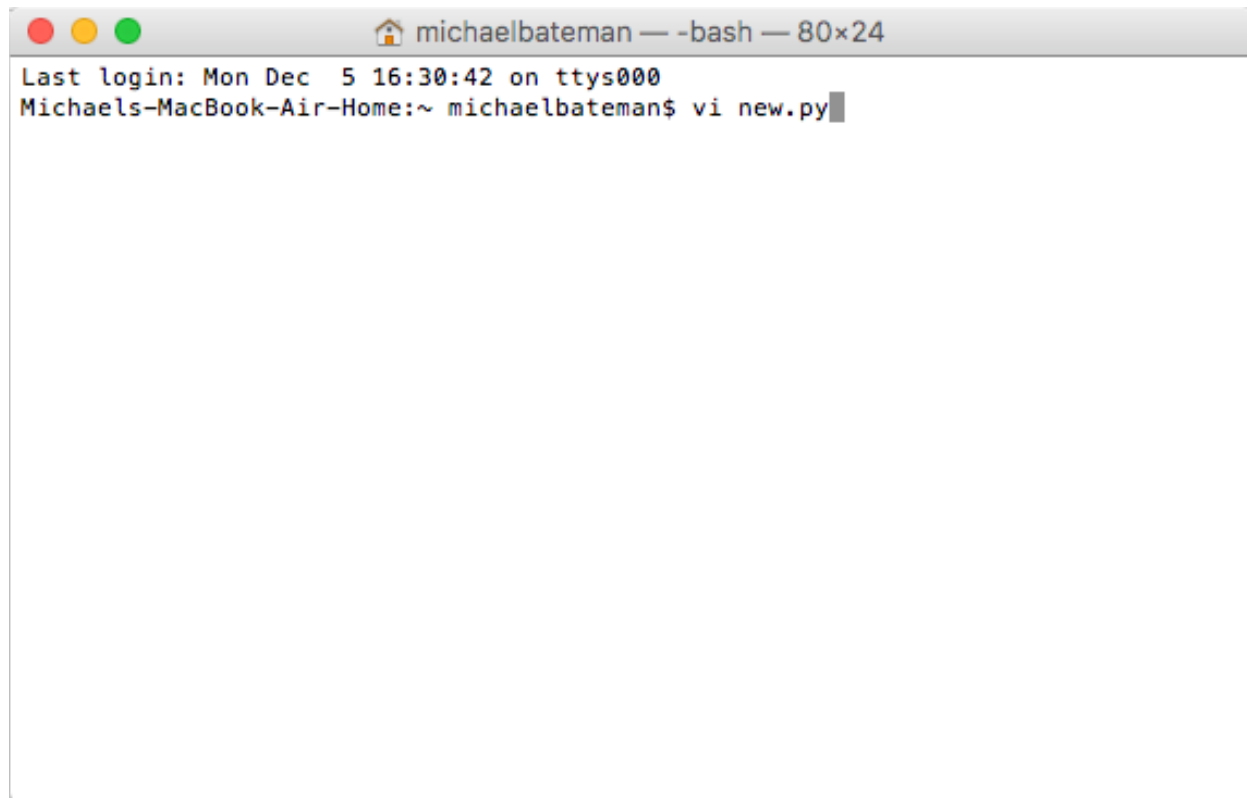
“:wq” - Write (save) and quit.

“yy” - How to yank (copy) a line of code.

“p” - How to paste the yanked text before the cursor.

“dd” - Deletes the line the cursor is on.

Python is a programming language that allows programmers to write fewer lines of code than C++ or Java. Guido van Rossum created the Python language. Python is pre-installed on the Mac and can be used through the command line.



```
michaelbateman — -bash — 80x24
Last login: Mon Dec 5 16:30:42 on ttys000
Michaels-MacBook-Air-Home:~ michaelbateman$ vi new.py
```

This creates a new python file named “new”. It also will open the file in vi.

## Functions

I use the “getRGB” function many times, over and over again. When I use a function over and over again, I do not need to write as much code.

michael.bateman — vi ~/Downloads/python-turtl...

```
#functions-----
def getRGB(michael):
    turtle.colormode(255)
    r = random.randint(0,255)
    g = random.randint(0,255)
    b = random.randint(0,255)
    michael.color(r,g,b)
```

1

### Definitions:

1 - Generates a random colour using the RGB colour values

```
def shape(michael):
    getRGB(michael)
    michael.begin_fill()
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)
    michael.end_fill()
```

2

2 - Creates the hexagon shape, using the “getRGB” function

```
def outline(michael):
    michael.color('#ffffff')
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)
```

3

3 - Creates the hexagon outline of the shape with a white outline. Note: I used a hex colour value.

```
def next(michael):
    michael.pu()
    michael.right(turn)
    michael.forward(move)
    michael.left(turn)
    michael.pd()
```

4

4 - Moves the turtle on to the next hexagon, in the same row

```
def nextrow(michael):
    michael.pu()
    michael.left(turn)
    michael.forward(move*3)
    michael.left(30)
    michael.forward(length)
    michael.left(hexturn)
    michael.forward(length*1.1)
    michael.right(180)
    michael.pd()
```

5

5 - Moves the turtle to the next row

```
def startpoint(michael):
    michael.pu()
    michael.setx(-60)
    michael.sety(-60)
    michael.forward(136)
    michael.left(turn)
    michael.forward(80)
    michael.right(turn)
    michael.pd()
```

6

6 - This sets the starting point for the hexagons.

```
def square(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)
    michael.pd()
    getRGB(michael)
    michael.begin_fill()
    for i in range(4):
        michael.forward(600)
        michael.right(turn)
    michael.end_fill()
```

7

7 - This makes the square, again, using the “getRGB” function.

```
def end(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)
    michael.pd()
```

8

8 - This moves the turtle (circle shape) to the coordinates (-300,-300).

## Variables

Variables I used

```
#Variables-----  
length = (30)  
hexturn = (60)  
move = (length*1.75)  
turn = (90)  
notimes = (2)  
michael = turtle.Turtle()  
#-----
```

The variables allow me to call on an item over and over again. If I did not use variables, I would need to write a lot more code. Also, variables are helpful, because if I want to change the length of the hexagon, I can do that where I define the variables.

Otherwise, without using variables, I would need to pick through my whole code to change something. Below is an example of a variable being called.

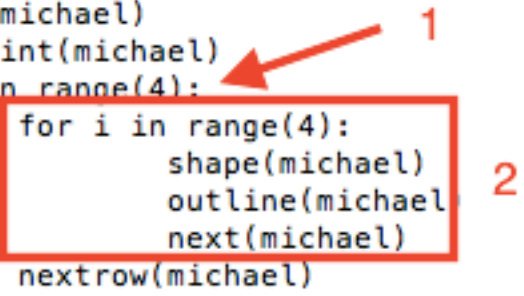
```
def next(michael):  
    michael.pu()  
    michael.right(turn)  
    michael.forward(move)  
    michael.left(turn)  
    michael.pd()
```

As you can see, I am using the variables that I defined, “turn” and “move”.

## Loops

Loops help me to repeat thing many times, without writing the code over and over again. Below, is an example of a loop in my code.

```
#rawcode-----
michael.shape("circle")
michael.left(turn)
square(michael)
startpoint(michael)
for i in range(4):
    for i in range(4):
        shape(michael)
        outline(michael)
        next(michael)
    nextrow(michael)
end(michael)
#-----
```




Definitions:

- 1 - This is a basic “for” loop.
- 2 - This is a “nested for loop”. Nested loops are loops inside of another loop.

## Imports

The import area is used to use other peoples sub-existing code.

```
#Import Area---
import turtle
import random
#-----
```



Definitions:

- 1 - This imports the python “turtle graphics”
- 2 - This imports random. Random is used when I get random RGB colour values.

## How my code works

```

#Turtle Art Python Test
#December 6, 2016

#Import Area---
import turtle
import random
#-----

#Variables-----
length = (30)
hexturn = (60)
move = (length*1.75)
turn = (90)
notimes = (2)
michael = turtle.Turtle()
#-----

#Functions-----
def getRGB(michael):
    turtle.colormode(255)
    r = random.randint(0,255)
    g = random.randint(0,255)
    b = random.randint(0,255)
    michael.color(r,g,b)

def shape(michael):
    getRGB(michael)
    michael.begin_fill()
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)
    michael.end_fill()

def outline(michael):
    michael.color('#ffffff')
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)

def next(michael):
    michael.pu()
    michael.right(turn)
    michael.forward(move)
    michael.left(turn)
    michael.pd()

def nextrow(michael):
    michael.pu()
    michael.left(turn)
    michael.forward(move*3)
    michael.left(30)
    michael.forward(length)
    michael.left(hexturn)
    michael.forward(length*1.1)

def nextrow(michael):
    michael.pu()
    michael.left(turn)
    michael.forward(move*3)
    michael.left(30)
    michael.forward(length)
    michael.left(hexturn)
    michael.forward(length*1.1)
    michael.pd()

def startpoint(michael):
    michael.pu()
    michael.setx(-60)
    michael.sety(-60)
    michael.forward(136)
    michael.left(turn)
    michael.forward(80)
    michael.right(turn)
    michael.pd()

def square(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)
    michael.pd()
    getRGB(michael)
    michael.begin_fill()
    for i in range(4):
        michael.forward(600)
        michael.right(turn)
    michael.end_fill()

def end(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)

#----- calls on variables -----
#rawcode-----
michael.shape("circle")
michael.left(turn)
square(michael)
startpoint(michael)
for i in range(4):
    for i in range(4):
        shape(michael)
        outline(michael)
        next(michael)
        nextrow(michael)
end(michael)
#-----
turtle.done()

```

In my code, I call on all of the functions that I made. Everything in my code is well used. You can find more information about the functions in the “function” area.

## Final Product

Code - without annotations (sorry, it could not fit on one image...)

```

#Turtle Art Python Test
#December 6, 2016

#Import Area---
import turtle
import random
#-----

#Variables-----
length = (30)
hexturn = (60)
move = (length*1.75)
turn = (90)
notimes = (2)
michael = turtle.Turtle()
#-----

#Functions-----
def getRGB(michael):
    turtle.colormode(255)
    r = random.randint(0,255)
    g = random.randint(0,255)
    b = random.randint(0,255)
    michael.color(r,g,b)

def shape(michael):
    getRGB(michael)
    michael.begin_fill()
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)
    michael.end_fill()

def outline(michael):
    michael.color('ffffff')
    for i in range(6):
        michael.forward(length)
        michael.right(hexturn)

def next(michael):
    michael.pu()
    michael.right(turn)
    michael.forward(move)
    michael.left(turn)
    michael.pd()

def nextrow(michael):
    michael.pu()
    michael.left(turn)
    michael.forward(move*3)
    michael.left(30)
    michael.forward(length)
    michael.left(hexturn)
    michael.forward(length*1.1)

def nextrow(michael):
    michael.pu()
    michael.left(turn)
    michael.forward(move*3)
    michael.left(30)
    michael.forward(length)
    michael.left(hexturn)
    michael.forward(length*1.1)
    michael.pd()

def startpoint(michael):
    michael.pu()
    michael.setx(-60)
    michael.sety(-60)
    michael.forward(136)
    michael.left(turn)
    michael.forward(80)
    michael.right(turn)
    michael.pd()

def square(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)
    michael.pd()
    getRGB(michael)
    michael.begin_fill()
    for i in range(4):
        michael.forward(600)
        michael.right(turn)
    michael.end_fill()

def end(michael):
    michael.pu()
    michael.setx(-300)
    michael.sety(-300)
    michael.pd()

#-----

#rawcode-----
michael.shape("circle")
michael.left(turn)
square(michael)
startpoint(michael)
for i in range(4):
    for i in range(4):
        shape(michael)
        outline(michael)
        next(michael)
        nextrow(michael)
    end(michael)
#-----

turtle.done()

```

## Drawing

