6. IMPORTING MODULES

When we are creating our programs we may need to use additional libraries of code which are not available as standard modules in the interpreter. Some common examples are:

- date and time
- math
- sys
- random

In order to use these additional modules we need to import them into our program by adding an import statement.

6.1 IMPORT DATE AND TIME

There are a number of options but these are likely to be more useful for programming project tasks.

- Date – manipulates dates as days, months and years.
- Time – measures time in hours, minutes or seconds.
- Datetime – combines dates and time.

```
>>>
Today's date : 2015-11-27
Today's date : 27-11-2015
I was born on a Saturday
>>>
```

```
import time
import datetime

today = datetime.date.today()

print("Today's date : {0}".format(today))

# Ideally I want to format the date as day,month,year

today = datetime.datetime.now().strftime('%d-%m-%Y')

print("Today's date : {0}".format(today))

#Find out the day of the week you were born

birthday = datetime.date(1999,4,10)
print("I was born on a {0}".format(birthday.strftime('%A')))
```


The date is displayed using the **strftime()** format and there are a number of additional 'directive' options for displaying dates and times:

strftime() directive	How it is displayed
%Y	Year, e.g. 2015
%y	Year, e.g. 15
%m	Month as a number, e.g. 11
%B	Month name, e.g. November
%b	Shortened month name, e.g. Nov
%d	Day in the month, between 01 and 31
%A	Weekday name, e.g. Monday
%a	Shortened weekday name, e.g. Mon
%p	AM or PM

We can also use the time.sleep() function to add a delay to our code by using the current system time on your computer.

```
import time

# create a countdown timer

delay = int(input("How many seconds delay?: "))

start = time.time()
finish = start + delay # the current time + the delay

x = 1
while time.time() < finish: # uses the current time on your computer
    print("{0}...".format(x))
    x += 1
    time.sleep(1) # delay for 1 second
print("Time's up!")
```

You can also add delays in your code by simply adding time.sleep() and the number of seconds delay you want before the next line of code is executed.

```
print("Hello")
time.sleep(2)
print("world!")
```

EXERCISE 10: USING DATE AND TIME

Write a program that will display the day of the week that you were born using the import time module.

You will need to use functions to ask for the date, month and year (as integers). You should display the date to the user entered in this format, e.g. 15/Jan/97, and tell them what day of the week they were born.

6.2 IMPORT MATH

You may need to use some more detailed mathematical functions in order to solve a problem; these are the common functions you may need in programming project tasks.

```
import math

# math floor returns the largest integer (regardless of 0.5 rule)

num = 105.25
print(math.floor(num))
print(math.floor(105.98))

# math ceiling returns the smallest (depending on the 0.5 rule)

print(math.ceil(num))
print(math.ceil(105.98))

num = 105.2575

# math pow raises the first of the arguments to the power of the second
# i.e. exponentiation 4**4 is 4 x 4 x 4 x 4 = 256

print(math.pow(4, 4))


# math sqrt returns the square root of the number e.g. √144 is 12

print(math.sqrt(144))

# round rounds down /up depending on the 0.5 rule

print(round(num))

print(round(num, 2))
```

```
math.floor(num)                    105
math.floor(105.98)                 105

math.ceil(num)                     106
math.ceil(105.98)                  106

math.pow(4 , 4)                    256.0

math.sqrt(144)                     12.0

round(num)-number of digits not specified          105

round (num, 2) - specified to 2 digits             105.26
```


6.3 IMPORT SYS

There is a useful function, sys.exit(), in the Sys module which can be used to exit out of your program code immediately rather than waiting for the code to finish.

This is used in this example code shown below so that the program does not return to the start of the while loop on line 14 when 4 has been entered as the menu_choice. The menu() function has already checked that the number selected is a valid choice so if the number entered is not 1, 2 or 3 then it must be 4.

```
11  def main():
12      """runs all the functions"""
13      menu_choice = 0
14      while menu_choice != 4:
15          menu_choice = menu()
16          if menu_choice == 1:
17              print("You have chosen to enter your name")
18          elif menu_choice == 2:
19              print("You have chosen to read the rules")
20          elif menu_choice == 3:
21              print("You have chosen to play the game")
22          else:
23              print("You have chosen to quit")
24              sys.exit()
```


6.4 IMPORT RANDOM

Sometimes you may want to choose a random item or sample of items from a list, or generate a random number from a given range.

Here are some examples:

```
import random

myList = ['ham','eggs','cheese','tomatoes','bread','butter']

for x in range (3):
    selection = random.choice(myList)
    print("The random selection is {0}".format(selection))


# get a random sample

lunch = random.sample(myList, 3)
space = ',' # print it out neatly
print("My random lunch consists of {0}".format(space.join(lunch)))

# generate a random integer between 1 & 5
num = random.randint(0, 5)
print("A random number between 1 and 5 is {0}".format(num))

#generate a random number, which may not be whole

num = random.random() * 100
print("A random number between 1 and 100 is {0}".format(num))

# generate a random number from a range (start,stop,step)

for i in range (3):
    print(random.randrange(0, 101, 5))
```

```
>>>
The random selection is ham
The random selection is butter
The random selection is cheese
My random lunch consists of cheese,bread,tomatoes
A random number between 1 and 5 is 4
A random number between 1 and 100 is 91.67714778689682
30
40
80
```

```
>>>
The random selection is bread
The random selection is ham
The random selection is ham
My random lunch consists of ham,cheese,butter
A random number between 1 and 5 is 5
A random number between 1 and 100 is 74.61959514012489
35
80
15
```


I can also use the random.randint() function to choose an item from a list using the index value. This shows an example of future-proofing your code, i.e. writing robust code. I may want to change my program later so I can add items to the menu; using the len() method on the list allows this to happen.

Example:

```
import random

menu = ['ham','eggs','cheese','tomatoes','bread','butter']

choice = random.randint(0,len(menu) - 1)

print(menu[choice])
```

EXERCISE 11: RANDOM OPTIONS

Write a program for a restaurant where the customers are given a random meal.

You will need lists for the starters, main course, desserts and drinks, with at least four items in each of your choices.

The program must generate a random meal with a choice from each menu option and display this to the customer like this:

```
>>>
************************************
                MENU
 Starter: Tomato soup
 Main : Fish and Chips
 Dessert: Mango Surpise
 Drinks: Cup of Coffee
************************************
```


6.5 IMPORT TURTLE

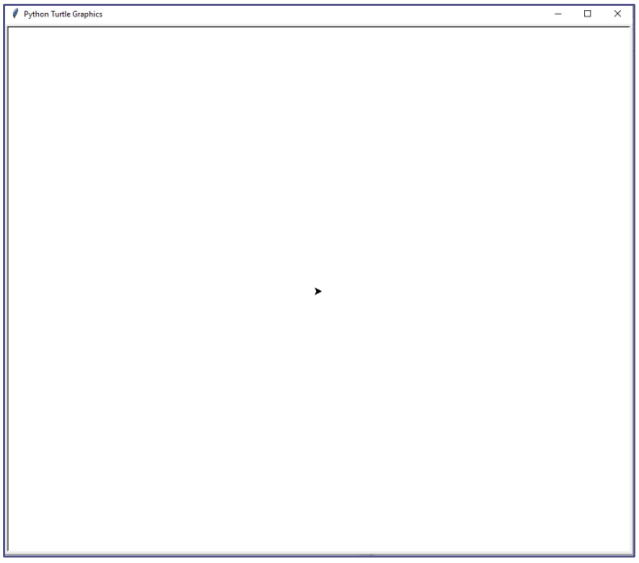
Just like the previous modules, the turtle module or library contains extra methods and functions. The main difference is that this is a GRAPHICAL library which is commonly used to help beginners understand some of the basic programming concepts in Python.

GETTING STARTED

1. Import the module

```
import turtle

# create the variables for the screen and the turtle

my_screen = turtle.getscreen()
terry = turtle.Turtle()
```

2. Create a variable for your drawing area – the screen and your turtle
3. Run the module and the screen appears with the 'turtle' (a black arrow) in the centre



The turtle now acts as a pen on the canvas (the screen); we can also change some of the features of the turtle:

- Color (note American spelling)
- Size
- Speed

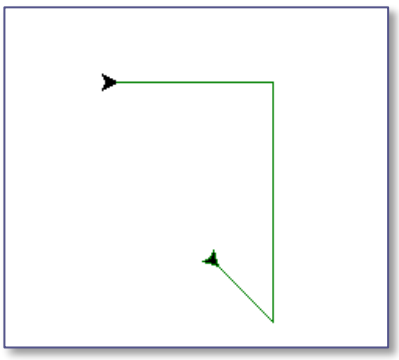
From its 'home' position in the centre of the screen, the turtle can move in four directions:

- Forward
- Back
- Left
- Right

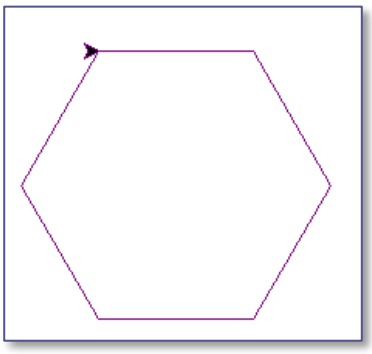
When moving forwards or backwards, you must specify the number of 'units' to be moved; when turning left or right, you must specify the number of degrees (clockwise or anticlockwise).

Simple example:

```
import turtle

# create the variables for the screen and the turtle

my_screen = turtle.getscreen()
terry = turtle.Turtle()


# change the pen color

terry.pencolor("green")

# make some basic moves

terry.forward(100)
terry.right(90)
terry.forward(150)
terry.left(45)
terry.back(50)
```



```
# change the pen color

terry.pencolor("purple")

# draw a hexagon

terry.forward(100)
terry.right(60)
terry.forward(100)
terry.right(60)
terry.forward(100)
terry.right(60)
terry.forward(100)
terry.right(60)
terry.forward(100)
terry.right(60)
terry.forward(100)
terry.right(60)
```

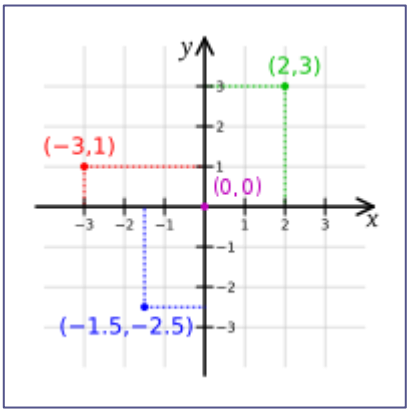


This can clearly be improved with the introduction of count-controlled iteration:

```
# change the pen color

terry.pencolor("purple")

# draw a hexagon

for x in range(0, 6):
    terry.forward(100)
    terry.right(60)
```

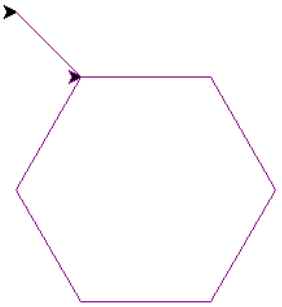

The screen uses Cartesian coordinates with the centre of the screen, where the X and Y axes cross, called the **Home**. This has the coordinates (0, 0) and the turtle always starts here.

The turtle can be moved anywhere on the screen using the command setpos() and the X and Y coordinates:



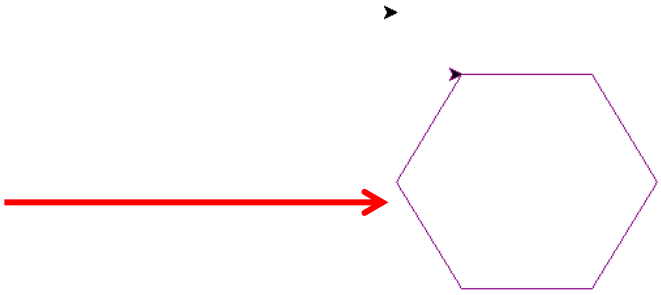

```
# draw a hexagon

terry.setpos(50, -50)

for x in range(0, 6):
    terry.forward(100)
    terry.right(60)
```

The resulting output highlights a problem to be resolved.



Moving the turtle to a new location also draws a line from Home to the new location. To solve this problem, lift the pen before the turtle moves and put it down again before the shape is drawn in the new location.

```
# draw a hexagon
terry. penup()
terry.setpos(50, -50)
terry.pendown()

for x in range(0, 6):
    terry.forward(100)
    terry.right(60)
```



EXERCISE 12: CREATE A SHAPE

Write the code to create:

- a regular octagon in red
- a regular nonagon in blue

In your answer:

a) Use condition-controlled iteration
b) Both shapes must be displayed on the same screen

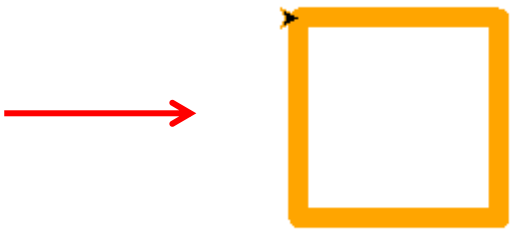
Note: use any suitable names for your screen and turtle variables.

Additional features include:

1) Clearing the drawing screen using the reset command
2) Changing the width of the pen

```
# draw a hexagon
terry. penup()
terry.setpos(50, -50)
terry.pendown()

for x in range(0, 6):
    terry.forward(100)
    terry.right(60)

terry.reset()

terry.pencolor("orange")
terry.pensize(10)

for x in range(0, 4):
    terry.forward(100)
    terry.right(90)
```

The hexagon is drawn first, then the screen clears and the orange square is drawn.



3) The turtle can also be visible or invisible; this is especially useful when drawing complex shapes as it speeds up the process

- hideturtle()
- showturtle()

```
t.pencolor("orange")
t.pensize(2)

x = -200
y = -200
c_size = 50

t.hideturtle()
for x in range (0, 12):
    t.penup()
    t.setpos(x , y )
    t.pendown()
    t.circle(c_size)
    x += 20
    y += 20
    c_size += 10
```