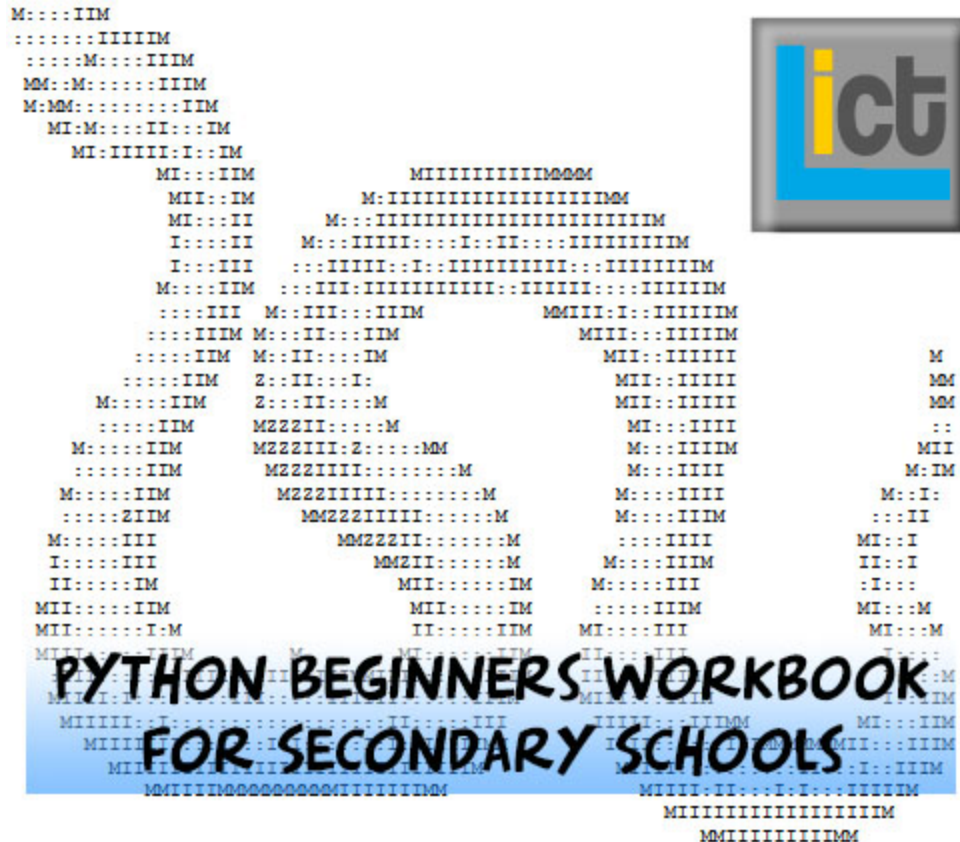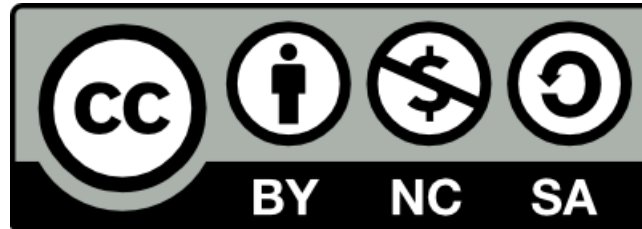# EXEMPLAR ANSWERS BOOK

Written by Ali Mulla

Licence

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

	Attribution — You must give the original author credit What does "Attribute this work" mean? The page you came from contained embedded licensing metadata, including how the creator wishes to be attributed for re-use. You can use the HTML here to cite the work. Doing so will also include metadata on your page so that others can find the original work as well.

	Non-Commercial — You may not use this work for commercial purposes.

	Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

With the understanding that:

- **Waiver** — Any of the above conditions can be **waived** if you get permission from the copyright holder.
- **Public Domain** — Where the work or any of its elements is in the **public domain** under applicable law, that status is in no way affected by the licence.
- **Other Rights** — In no way are any of the following rights affected by the licence:

Contents

Introduction

This book aims to be used as a workbook to introduce students to programming in the Python programming language in a simple manner.

I find that younger learners can often find concepts hard to grasp if the concept is merely lectured to them.

The aim of the book is to avoid teaching any concepts too complicated for the time being and teach young learners in the same way they may learn other concepts in a secondary school using exemplary and repetitive exercises.

It's important to note that this book is not meant to replace the teaching of concepts in the classroom but to help reinforce some of the topics with worksheets that students can use to practice the concepts that have been discussed by a teacher. You will find on many of the worksheets that to avoid having too much text, I have left out some explanations that may involve giving the student too much information and detract from the practicing the new skill.

In essence, computer programming is all about solving problems. The aim of this book is to help teach the basic syntax and concepts of Python before presenting students with more complicated problem solving tasks later on (probably to be presented in book 2 😄).

If you like this book, please share it with colleagues and peers. If you find errors or you have suggestions for improvement, please contact me via the following channels:

The contact form on the bottom of the homepage www.learnICT.it
Twitter handle: LearnICTit

Hanging words off a string - Task 1

In Python, getting the computer to write words can be done using the **print** function.
For example the code would look like this:

```
print("Hello World")
```

The word **print** tells Python that you want it to print some words out on the screen.
This is called printing a string. Imagine letters being stringed up back to back and displayed on the screen in the order they have been *strung up*!

In between the brackets we also need to add what we want Python to print in quotation marks.
So I could write:
        "My name is He-Man"
Like this:

```
print("My name is He-Man")
```

Did you notice how the print function is written in lowercase?
Write your name in the box below using the Python language.

```
print("name")
```

Write another line in the Python language stating who your favourite singer or band is.

```
print("Michael Jackson is my favourite singer")
```

What do you think would happen if the Python code you wrote included quotation marks?
For example:

```
print("She said "she loves you" and you know you should be glad")
```

Try this out in Python and write your answer in the box below.

```
print('She said "she loves you" and you know you should be glad')
```

the print function will stop printing out the rest of the line after the second double quote. Pytho
can also use single quotes in this case. See example above.

Hanging words off a string - Task 2

Python can also use single quotation marks when using the print function to write text on the screen. For example:

```
print('my name is He-Man')
```

By using a combination of both we can write sentences that need to use quotation marks.

Try to see if you can see what is missing from the following lines of code and write in the correction underneath.

1 - print(hello world)

print("hello world") - missing quotes

2 - (print"how tall are you?")

print("how tall are you?") - must go outside of brackets

3 - ('Game Over')

print('Game Over') - missing print

4 - print('Game Over")

print('Game Over') - mixed quotation symbols used. Stick to either single or double

5 - print('Bob's car has 4 wheels')

print("Bob's car has 4 wheels") - Apostrophe (single quote) after Bob stops the rest of the line from printing. Use double quotes to start and end the print function.

John Motson style lines of commentary

OK, this has nothing to do with football! When writing code, you should add comments to your work so that you can remind yourself later what a section of code means. Sometimes looking at lots of code can be complicated so it may be useful to add descriptive comments to help yourself and others who may be reading your code make sense of it.

A line of commentary is a line of text that the computer will not read as code. The computer will ignore the line and move to the next line of code.

There are 2 ways to add comments:
1. For a single line of commenting you simply add a hash in front of the line that will be commented.
2. For comments that will be written over several lines, you can add 3 quotation marks before and after your comments. This can be either single or double quotation marks ("' or """)

Type the following code into Python and see what happens when you try to run it:

```python
# This line of text is ignored. The next line prints some text
print("My name is...")

# This line is ignored
print("My name is...")

'''
When you need to have
several lines of commentary
like this, you can use 3
quotation marks.
'''
print("Slim Shady")
```

Although there will not be many lines of commentary in this book, you should make a habit of adding them to each line of your code like the one above to explain what each code does.

Python can do maths!

Python can be used to work out maths problems. You can test this out just by typing equations into Python and seeing for yourself.

Try the following:
2 * 2
2 + 2
2 / 2
2 - 2

The +, - / and * symbols are called operators because they tell the computer to do an operation. try the following in the Python interpreter and write the answers next to the equations.

1. 12 * 12
2. 1 + 78
3. 45 / 3
4. 10 * 10
5. 10 / 2
6. 10 + 10
7. 10 – 2
8. 10 + 10 + 10
9. 10 * 2 + 5 / 2 - 2

In Python an **integer** is a whole number such as 5, 10, 123 or 7.
A floating point number is a number with decimals such as 1.1, 0.5 or 2.99.
A number with a decimal point of 0 such as 2.0 is still considered as a whole number so is an **integer**.

Use the Python interpreter to work out these equations:

1. 2 + 5.6
2. 2 * 5.6
3. 12 / 1.2
4. 7.8 - 4
5. 15 / (1.5 * 3)

Concatenation is a hard word to remember!

So you have now seen that Python can be used to work out equations, we will now ask Python to print numbers and words together. In programming languages adding together strings with numbers or strings with more strings is called concatenation.

For example I could say "I am a school student and I am aged 14"

```
print("I am a school student and I am aged ", 14)
```

Or I could write:

```
print("2+2 = ", 2+2)
```

Try the following concatenations in the Python interpreter and write in the missing code.

> Concatenating an integer along with a string can be done by wrapping the integer with str().
> This basically converts the integer and makes it suitable to print alongside a string.

1 - print("hello " + "world)

> print("hello " + "world") - missing a quotation mark at the end

2 - print("I am Jerry, and I am ", 14 years old")

> print("I am Jerry, and I am ", 14, " years old") - Missing comma to concatenate and quotation symbol.

3- print('I'm Jerry and I'm' + 14 + 'years old')

> print("I'm Jerry and I'm" + 14 + 'years old') - Single quotes used in a sentence with an apostrophe.

4- print("if I had 1 apple and 2 pears, I would have " + str(2 + 1) fruit)

> print("if I had 1 apple and 2 pears, I would have " + str(2 + 1) + "fruit")
> Must concatenate the last word too.

5- print("if I had ", 1, "apple and ", 2 "pears, I would have ", 2 + 1, "fruit")

> print("if I had ", 1, "apple and ", 2, "pears, I would have ", 2 + 1, "fruit")
> Missing a comma to concatenate the second integer.

Variables - Task 1

In computing terms a variable is something that the computer has to remember. It can refer to it and can make changes to it, if you ask it to. Sometimes people refer to a variable as a nickname for something. In a game, variables are usually the things can change such as the score, the timer or the lives of a player.
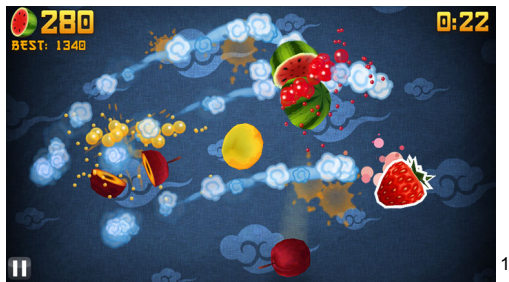
In the example above, you can see the variables are score, timer and best score.
See if you can spot the variables in the following images:

Image	Variables
	❏ Score ❏ Coins ❏ World ❏ Timer ❏ Character
	❏ Score ❏ Highest score ❏ Health Bar (graphical) ❏ number of KO's ❏ Timer ❏ Character name

---

[1] Fruit Ninja by, © Halfbrick http://fruitninja.com/
[2] Mario Bros © by Nintendo
[3] Street Fighter 2 © by Capcom

Setting a variable is easy in Python. Whenever any word or number follows a **=** symbol, it is regarded by Python has a variable.

For example to set a variable called bandName to The Beatles we would do this:

bandName = "The Beatles"

Now we can concatenate the variable with strings.

bandName = "The Beatles"
print("The most successful band ever were " + bandName)

Try these out and see if you can correct the errors

1 -bandName = "The beatles"
   print(bandName " had 27 number 1 singles")

---

print(bandName **+** " had 27 number 1 singles")
Missing **+** for concatenation

---

2 - bandName = "The beatles"
   print("bandName", are the most successful band ever with 15 number 1 albums)

---

print(bandName, **"**are the most successful band ever with 15 number 1 albums**"**)
Quotes are around variable instead of string.

3- Complete the following code:

bandName = "The beatles"
nickName = "The Fab Four"
print(bandName + "were also known as _____)

---

print(bandName + "were also known as**" + nickName + "."**)

We can change variables by reassigning the variable as something else. for example:

batman = ~~"Adam West"~~
batman = ~~"Michael Keaton"~~
batman = ~~"George Clooney"~~
batman = "Christian Bale"

Every time the variable batman is reassigned as something else it forgets the old name. There is no way to bring that back unless it is reassigned as a previous value.

Use the Python interpreter to :
  1. Assign Christian Bale as the variable for batman.
  2. Print the variable for batman
  3. Assign Ben Affleck
  4. Print the variable for batman

Variables - Task 2

Don't forget a variable is just something the computer has to remember. It may have to remember many variables. Although variables are not just used in games, games are a good example of how they are used.

For this example we will use a parable of a famous Italian plumber in order to not upset a large corporation.

Meet Mahmud the fez wearing Turkish Mechanic. He has 3 lives. If he touches alcohol, he loses a life but if he touches lokum (Turkish delight), he gains a life.	lives = 3 print(lives) If Mahmud were to touch lokum, I would have to change his life like this: lives = lives + 1

Try this out in Python

```
lives = 3
print(lives)
lives = lives +1
print(lives)
```

1 - What would you have to type in to reduce the lives by 1? Try it out and write the correct answer below.

> **lives = lives - 1** (or alternatively: **lives -= 1**)

2 - Mahmud jumps on a lives bonus which gives him 3 lives at once. How would the code to make his life go up look for this?

> **lives = lives + 3** (or alternatively: **lives += 3**)

Comparative Operators - Task 1

In programming we often ask computers to make checks to see if something is true or false. This helps computers make decisions based on the algorithms we write. Take for example our game hero Mahmud, if his lives dropped to 0 then the game would finish and display the game over screen:

```
lives = 1
if lives == 0:
    print("Game Over!!")
```

You would have learned about comparative operators in maths lessons. Try these out in the Python Interpreter and write the answers down. Python will reply either True or False:

1. 3 < 1

False

2. 3 > 1

True

3. 3 == 2+1

True

4. 5 -2 > 1

True

5. 1 == 1

True

6. 0 > 0.001

False

In Python remember that adding an = symbol creates a variable. In order to test whether something is equal to something we use ==.

User Input

We have looked at how to write text and how to create variables but now we need to get the computer to ask questions and save the answer we give.

Python uses the input() function to ask a question to the programme's user.
For example:

| input("What is your name?:   ")
print("my name is R2D2") | The first line asks for your name.
The 2nd line merely states a name. |

The problem with the above code is that after asking for your name it doesn't save it. Do you remember that by using a = symbol we could create a variable?

| yourName = input("What is your name?:   ")
print("my name is R2D2")
print(yourName + " was my dad's name") | Now the computer remembers the name you give and can use it. |

1. Get Python to ask you your name. Python must reply to you and say hello to you while referring to you using your name. Write the code into box below.

```
yourName = input("What is your name?:   ")
print("Hi " + yourName)
```

2. Now get Python to ask you your name and age. Python must reply hello yourName and tell you that you are the same age. Write the code into box below.

```
yourName = input("What is your name?:   ")
print("Hi " + yourName)
yourAge = input("What is your age?:   ")
print("Oh! I am also " + yourAge + "years old")
```

Comparative Operators 2 - If Statements

Your task now is to combine a comparative operator with an input. Computers often use if statements to determine what decision to make. We touched on this earlier with Mahmut who lost a life when he touched alcohol.
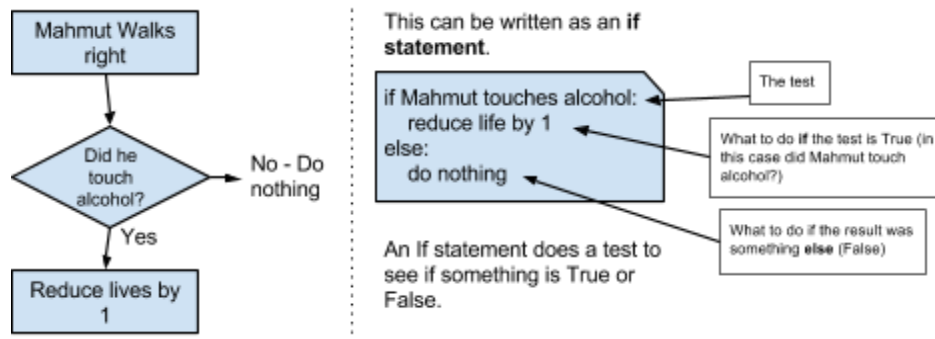
Task 1. Let's write a basic if statement using a variable first.

age = int(input("What is your age?"))
if age < 16:
    print("you are young!")
else:
    print("your are getting old!")
	Re-write this if statement to ask how many marbles I have. (Notice how the code **int** is infront of **input** because the data being entered is a number.)
	
	If the person has less than 6 marbles, it should reply "I have more than you"
	If the answer is else, it should reply, "Wow that's a big collection of marbles!!"

Write your answer below:

```
marbles = int(input("How many marbles do you have?"))
if marbles < 6:
    print("I have more than you")
else:
    print("Wow that's a big collection of marbles!!")
```

Task 2. New challenge. Let's get Python to ask if my name is John. If it is, Python should reply, "That's my name too!" if else Python should reply, "Well it's not a great name but it will do".

```
name = input("What is your name?:   ")
if name == "John":
    print("That's my name too!")
else:
    print("Well it's not a great name but it will do")
```

Remember the == operator checked if something was the same as something else.

Comparative Operators 3 - More If Statements

If statements can check for a list of things instead of just 1. For example:

```python
temp = int(input("What's the temperature in celcius?"))

if temp < 8:
    print("Brr, its a cold day!!!")
elif temp >= 8 and temp < 15:
    print("I'ts a mild day")
elif temp >= 15  and temp < 21:
    print("Its a warm day")
elif temp >= 21:
    print("It's a hot day")
else:
    print("sorry, you must enter a number")
```

Can you see that the word **if** is only used once?

Every other time to check if something is **True**, Python uses **elif**.

Task 1
Write a programme to check if someone's name is John, George, Ringo or Paul. If one of those names is True, Python should say "Hey that's the name of a Beatle!".
If not, Python should say "That's a nice name".

Write your answer in the next box

```python
name = input("What is your name?:   ")
if name=="John":
    print("Hey that's the name of a Beatle!")
elif name=="Ringo":
    print("Hey that's the name of a Beatle!")
elif name=="George":
    print("Hey that's the name of a Beatle!")
elif name=="Paul":
    print("Hey that's the name of a Beatle!")
else:
    print("That's a nice name")
```

Task 2
Write a programme to comment on how interesting a football match was.
  ❏ If no goals were scored, Python should say "the game was a bore draw!"
  ❏ 1-2 goals: "Not the most interesting game"
  ❏ 3-5 goals: "It was a very interesting game"
  ❏ 6+ goals: "The football match was an unmissable game!"

Write your answer in the next box

```python
goals = int(input("How many goals were scored in the game?:"))
if goals==0:
    print("the game was a bore draw!")
elif goals > 0 and goals <=2:
    print("Not the most interesting game")
elif goals >= 3 and goals <=5:
    print("It was a very interesting game")
else:
    print("The football match was an unmissable game!")
```

Datatypes

Programming involves asking a computer to work with different types of information. For example numbers and words. We will focus mainly on 3.

Python understands that numbers are different than strings. For example it can work out an equation with numbers like 2 * 2 but not with strings like word * word. There are exceptions to this and you will see.

Task 1

Use Python to tell us what type of information each of these are:

print(type(10))

```
<class 'int'>
```

print(type(10.5))

```
<class 'float'>
```

print(type('letters & words'))

```
<class 'str'>
```

Task 2

Try these in Python too using **print(type())**:

34, "phone", 99.9, "The Hobbit", 95.8, 12, 16, 1, 0.5, 0.9, and 12.45.

Task 3

Can you understand the 3 different types of data? Try to explain what the difference between an integer, a float and a string is.

❏ Integer is a whole number
❏ a float is a number with more than 1 decimal point
❏ a string can contain data that is letters, symbols and numbers.

Your teacher will discuss the answers with you.

Looping - While Loops

Computers can be programmed to continue repeating code *while* a condition is True.
Look at the example below for an idea.

| ```python
name = ""
while name != "Batman":
    print("Somebody call Batman!")
    print("Are you him? ")
    name = input("What's your name?")
``` | As you can see in the example in the left, the code will keeping looping and asking the same question **while** the answer is not Batman.<br><br>Notice the nesting of statements underneath the while? It is the indented code that will keep looping. |

Task 1

Write a while loop to ask someone what the best football team in the world is. Repeat the question until they say the correct team. Test it in Python and write your answer and explanation in the table below. Share the code with your teacher and class.

| ```python
team = ""
while team != "Galatasaray":
    print("The best team in the world are")
    print("GALATASARAY ")
    team = input("What's your favourite
team?")
``` | |

Task 2

Try to write your own infinite while loop like the one above. This is a hard task but you can be creative and make code for whatever you want. Test it in Python and write your answer and explanation in the table below. Share the code with your teacher and class.

Task 3

Now we will try to write one more **while** loop this time using a condition. Again this task is a lot harder. Look at the example below and try to write your own.

| ```python
secret_number = 6
guesses = 3
while guesses > 0:
    print("Can you guess what number I am thinking?)
    guess = int(input("write a number "))
    if guess == secret_number:
        print("Well done, you guessed correctly!")
        break
    else:
        guesses = guesses -1

if guesses == 0:
    print("You have no guesses left!")
``` | This is a little more complicated. it may be a good idea to write this out as a flowchart to help you think about how the code works.<br><br>In this code, we also learn of the **break** statement. This will tell the PC to skip to the next line of code that is not in the current indented nesting.<br>(Alternatively we could replace break with guesses = 0) |

Task 4 (A little harder :-/)

The code below shows an example of how to add a list of numbers together:

```python
count = 5
number = 0
print("this programme will work out the total of a list of numbers")
while count > 0:
    number = number + int(input(print("Please enter a number :")))
    count = count - 1
print(number)
```

Edit this code in the box below to also divide the total by the count.

```python
count = 5
number = 0
print("this programme will work out the total of a list of numbers")
while count > 0:
    number = number + int(input(print("Please enter a number :")))
    count = count - 1

total_dev_count = number / count
print(number)
print(total_dev_count)
```

Try it in Python and share your answer with your teacher and class.

Inserting Modules

While Python includes a lot of functions like **print**, it leaves out a lot of functions that we can load into our code if we need it. Think of a module as a book with instructions on how to do something. These modules are stored by Python in a library until we need our code to know how to use them.

The code below shows an example of how to add actual time to our code.

import time
 	
count = 10
while count >0:
   print(count)
   count = count - 1
   time.sleep(1)
print("Time is up!")
	This line imports the module
	
	In the code, you can see **sleep**. A function within the **time** module which pauses the code for the amount of time within the brackets.

Task 1

Create a code to count from 0 to 10. Write the answer below and share with your teacher. You will need the sleep function.

TASK 1	**TASK 2**

import time
 	
count = 0
while count <11:
   print(count)
   count = count + 1
   time.sleep(1)
print("You have had 10 seconds!")
	from time import sleep
	
	count = 0
	while count <11:
	   print(count)
	   count = count + 1
	   sleep(x)
	print("You have had 10 seconds!")

Task 2

When importing modules into your code, Python adds many functions into the memory. This could slow down how fast it takes to execute your code when it's time to run it.

Within the time module, there are loads of functions that we didn't use such as *clock*. We can choose to load just the function we need from a module like this:

from time import sleep

In order to use the function such as sleep, it now needs to be written into the code differently.

How it was written before (x is represents seconds)	How it should be written now
time.sleep(x)	sleep(x)

Re-write your code to reflect the changes.

Variables task 3 - Using Variables for Calculations

By now we know that variables can represent both numbers and text. We shall try to use variables to store formula to make it easy to calculate equations.

Area of a square is height x length $area = x \times y$	`height = int(input("What the height of your rectangle in CM?"))`
	`length = int(input("What the length of your rectangle in CM?"))`
	`area = height * length`
	`print("the area of the rectangle is " + str(area) + "cm\xb2")`
	
	`#Note that the code: \xb2 is an escape code which we will cover later.`

Use the example above to write a Python script to work out the area of a triangle asking the user for the height and width.

Area of a triangle is base x height ÷ 2 $area = x \times y \div 2$	`height = int(input("What the height of your triangle in CM?"))`
	`length = int(input("What the length of your triangle in CM?"))`
	`area = height * length / 2`
	`print("the area of the triangle is " + str(area) + "cm\xb2")`

Now use the example above to write a Python script to work out the area of a circle by asking the user for the radius. This is quite a hard task but you should be able to work it out if you could work it out on a calculator. The first variable has been written for you :-)

Area of a circle is Pi x radius x radius πr^2	`r = int(input("What the radius of your circle in CM?"))`
	`pi=3.14`
	`area = pi*r*r`
	`print("the area of the circle is " + str(area) + "cm\xb2")`

Defining a function - Task 1

When writing code sometimes we need to repeat code several times and therefore end up writing the same thing repeatedly. To cut down on this most programming languages allow you to create functions that allow you to call the function that performs an action when you need it.

To give you a basic idea we shall create a number machine that will receive a number and perform an action to it.

As you can see from the illustration on the left, the number 5 is given to the machine as an 'argument'.

It is then multiplied by 5 and divided by 2.

The code for such a function could look like this:

def numberMachine(x):
    x = x * 10 / 2
    print(x)
 	
numberMachine(5)
	Here the name of the function is defined. x will be replaced with a number.
	Underneath the name of the function the calculation is added.
	
	Wherever the name of the function is written on a page of code, it will 'call upon' it and execute it. This can be called *calling the function*.

Task

1. Write a number machine function to take a number and multiply it by the power of 3 & divide it by 3. (**easy**)	2. Write a function to take the radius of a circle and work out the area. (**hard**, refer to variables task 3)

def numberMachine(x):
    x = x * x * x / 3
    print(x)
 	
numberMachine(x)
	def numberMachine(x):
	    pi = 3.14
	    y = pi * x * x
	    print(y)
	
	numberMachine(x)

Defining a Function - Task 2

This task is in essence the same as the previous task but offers more practice.

Task 1

Write a function called **tax** that takes the cost of a meal and adds on 22% VAT. This can be worked out in many ways but you can refer to the equation below for help with adding VAT onto any number (x).

$$x + VAT = x \div 100 \times 122$$

Write your function below:

```
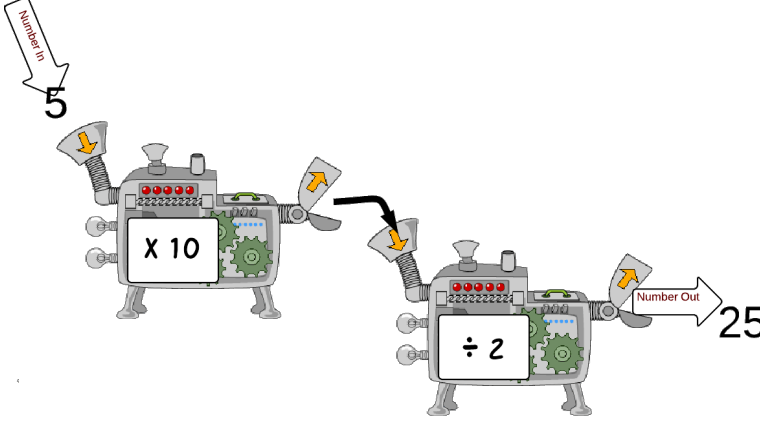def tax(x):
    vat = x * 0.20
    print(x + vat)
```

Task 2

Write a function called **warningMessage** to print out the phrase:
"*Danger Will Robinson \n*" the number of times of the number entered when the function is called. eg. If the number 3 is entered, it should print the phrase 3 times.

Write your function below:

```
def warningMessage(x):
    message = "Danger Will Robinson \n"
    print(x * message)

warningMessage(3)
```

Lists

A list in Python is a container that holds a list of objects (words or numbers). For example below is a shopping list for things that need to be bought.

shopping = ['bread', 'cola', 'shampoo', 'donuts', 'cheese']

To create a list give it a name (the same way you did for a variable) and put items of the list in square brackets separated by commas.

Task 1
Create a list called avengers for The Avengers characters, Captain America, Thor, The Hulk and Iron Man.

```
theAvengers = ['Captain America', 'Thor', 'The Hulk', 'Iron Man']
```

Try to print the list in Python by typing:
print(avengers)

In programming languages counting usually starts from 0. Therefore the first item in the list is 0 and the second item is 1.
We can print out single items from a list by typing the list name followed by the item number in square brackets.
print(shopping[0])

Task 2
a. Write the code to print the third string in the avengers list.

```
print(theAvengers[2])
```

b. Write the code to concatenate: "The character who has a special hammer is " with the 2nd string in the list.

```
print("The character who has a special hammer is " + theAvengers[1])
```

c. Write the code to concatenate that your favourite characters are the 1st and 4th in the list.

```
print("My favourite Avengers are "+ theAvengers[0] + " and " + theAvengers[3])
```