

# KS3 Programming Workbook

## INTRODUCTION TO



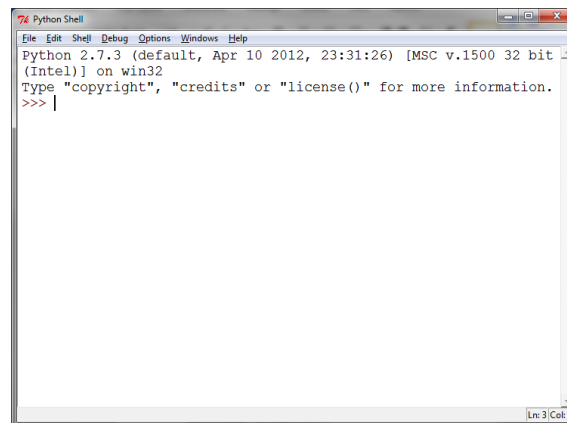
Name:

Class:

## Introducing Python

Python is a programming language that is easy to learn. It provides a way to write instructions that are simple for a human to understand and clear for the computer to follow.

Open up the Python IDLE. You should see this:

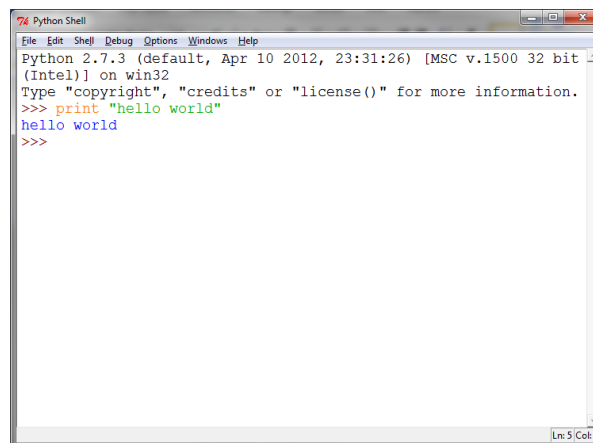


```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

This is the **Shell**, which waits for you to type in instructions at the `>>>` which is called the **prompt**.

Type in `print "hello world"`

You will need to type it in exactly as you see it here – don't put a P instead of a p. Hit **Enter** and you should see this:



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "hello world"
hello world
>>>
```

Try out a few more print statements. The computer will print out exactly what you put between the " and ".

Type in `print 3+4`. What happens?

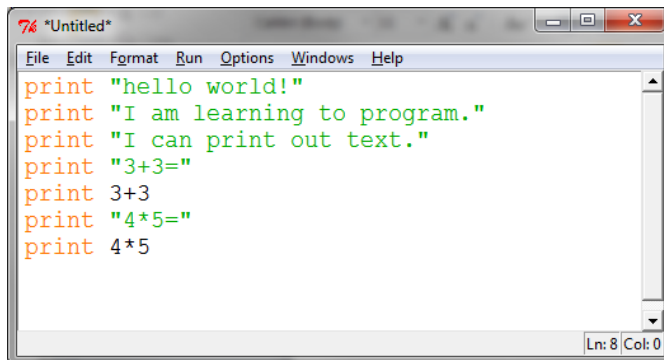
Type in `print 5*7`.

## Writing a program

So far you have given Python instructions one line at a time – that’s like you being sent to the shop for one item, then going home again to find out what the next item is! Now we need to learn how to write a list of instructions and give them all to Python at once.

On the Shell, click on **File/New Window**. You should see a completely empty window.

Type your code in here. Start each instruction on a new line.

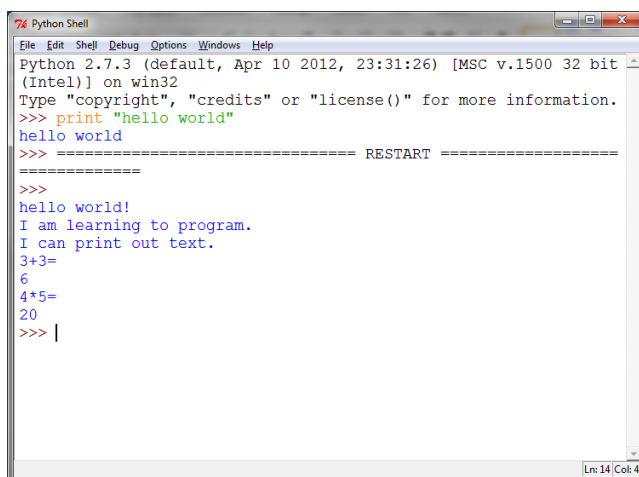


```
print "hello world!"
print "I am learning to program."
print "I can print out text."
print "3+3="
print 3+3
print "4*5="
print 4*5
```

Python colours your code so that you can understand it more easily. Anything in green is a **String**, which means Python treats it as text and will print it exactly as it appears.

Click on **File/Save as** and save your code with the filename **test.py**.

Click on **Run/Run Module**. You should see your statements appear in the Shell window.



```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "hello world"
hello world
>>> ===== RESTART =====
>>>
>>> hello world!
I am learning to program.
I can print out text.
3+3=
6
4*5=
20
>>> |
```

**Did it work?** Congratulations, you’ve written your first Python program!

**Problems?** If your program didn’t run at all, check very carefully that you have typed it correctly and followed all instructions. If your program runs but the colour coding in the listing disappears, resave your program with a different name and make sure you put **.py** on the end. This tells the computer that it is a python program and that it should be colour coded to help you.

## Programming time 1!

Now that you can write your very own program, write a program to solve these problems. Write down your answers. Can you explain what it does? Some are easy, some trickier.

Expression	Result	What does it do?
3+4		
57-4		
4*6		
7/3		
7%3		
2**3		
3==3		
3==6		
4<6		
4>6		
4!=6		
range(100)		

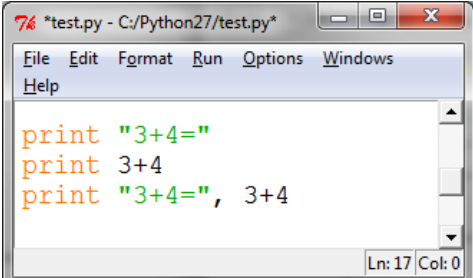
**Hint:** put a print command in front of each of these. You will find it useful to put a string in front of it to explain what it does as well.

For example:

```
print "3+4="
print 3+4
```

Or you could put them on the same line by using a ,

```
print "3+4=", 3+4
```



```
*test.py - C:/Python27/test.py*
File Edit Format Run Options Windows
Help
print "3+4="
print 3+4
print "3+4=", 3+4
Ln: 17 Col: 0
```

## Input in Python (numbers)

In order to write a useful program, we need to be able to ask the user questions. This means we need to store their answers as well.

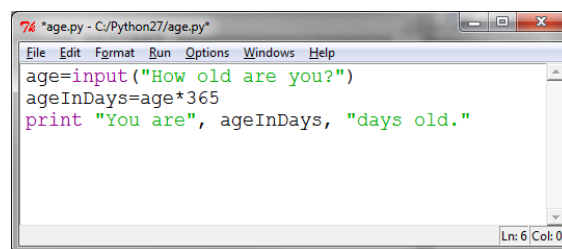
We can ask someone their age and store it as a **variable** called age like this:

```
age=input("How old are you?")
```

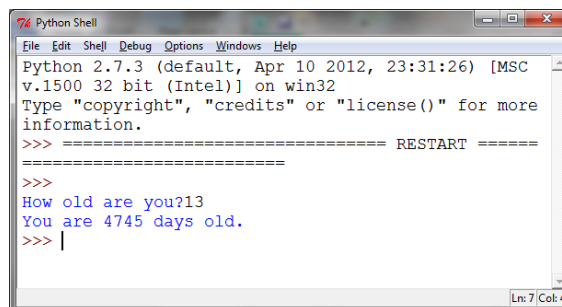
We can then do things with **age**, like multiply it by 365 to find out your age in days.

```
ageInDays=age*365  
print "You are", ageInDays, "days old."
```

As we're not allowed spaces in a variable name, we run words together and capitalise the first letter of each word. This is called camel case (full of humps!).



```
*age.py - C:/Python27/age.py*  
File Edit Format Run Options Windows Help  
age=input("How old are you?")  
ageInDays=age*365  
print "You are", ageInDays, "days old."
```



```
Python Shell  
File Edit Shell Debug Options Windows Help  
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC  
v.1500 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more  
information.  
>>> ===== RESTART =====  
>>>  
How old are you?13  
You are 4745 days old.  
>>> |
```

## Programming time 2!

1. Write a program to work out your age in hours ( $\text{age} * 365 * 24$ ).  
*The age 13 should produce 113880.*
2. Write a program to ask for width and height and print out the area of a rectangle ( $\text{width} * \text{height}$ ).  
*Width 5 and height 2 should produce area 10.*
3. Write a program to ask for time and speed and work out the distance travelled ( $\text{time} * \text{speed}$ ).  
*Time 2 and speed 60 should produce distance 120.*

Make sure that each time you choose a sensible name for your variables.

## Input in Python (Text)

If we want to input text instead of numbers, we need to use `raw_input( )` instead of `input( )`.

```
name=raw_input("What is your name?")
print "Hello", name
```

```
food=raw_input("What is your favourite food?")
print "I like", food, "too."
```

Write your own program to respond to the user.

## True and False in Python

You can test something to see if it is true. Python will respond with either True or False.

```
print 3==5
print 5>4
print 2<7
print 2>7
```

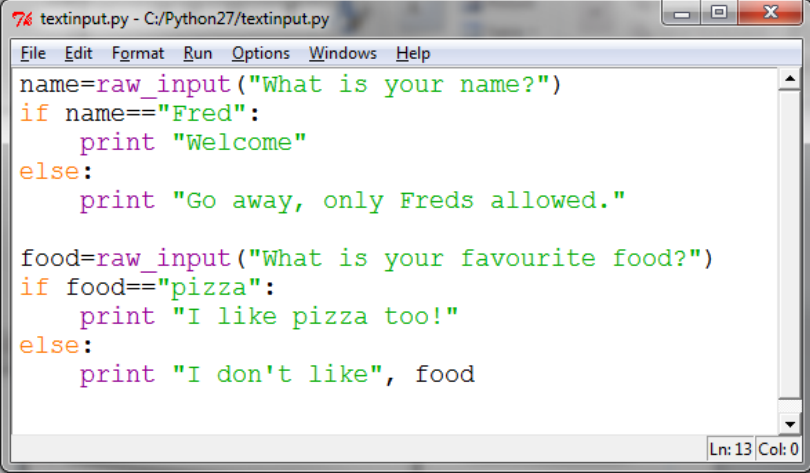
Make up some of your own statements and make sure Python answers right.

## Making choices in Python

We can tell the computer to check something and do something different if it is true or if it is false, by using an IF statement.

It will only carry out the indented instructions after the `if` line if the test is true.

You can also include an `else:` section to tell it what to do if the answer is false.



```
7% textinput.py - C:/Python27/textinput.py
File Edit Format Run Options Windows Help
name=raw_input("What is your name?")
if name=="Fred":
    print "Welcome"
else:
    print "Go away, only Freds allowed."

food=raw_input("What is your favourite food?")
if food=="pizza":
    print "I like pizza too!"
else:
    print "I don't like", food
Ln: 13 Col: 0
```

Look very carefully at the layout of the code, including where `:` is used and how the code is indented. Also note that we use `==` (two equals signs) to check if the variable is equal to what we want.

We can use IF statements in the same way with numbers.

```

File Edit Format Run Options Windows Help
age=input("How old are you?")
if age>=18:
    print "You are old enough to vote"
else:
    print "You are not old enough to vote yet."
Ln: 7 Col: 0
    
```

`>=` means more than or equal to.

In maths we would write  $\geq$

### Programming time 3!

1. Write a program to have a conversation with the computer. It should ask questions and then answer differently according to the answers given. Test your program out on a partner. Does it work?

Remember we use `raw_input( )` with text.

2. Write a program to have the user input a price. You should then check whether the price is less than 10. If it is, then print "You can afford it." Otherwise, print "It is too expensive."

Remember we use `input( )` with numbers.

<code>==</code>	is equal to
<code>!=</code>	is not equal to
<code>&lt;</code>	is less than
<code>&gt;</code>	is more than
<code>&lt;=</code>	is less than or equal to
<code>&gt;=</code>	is more than or equal to

We can add more choices by using `elif` – this is short for ELSE IF and we need to do another test with it.

```

if number>50:
    print "Too big"
elif number < 10:
    print "Too small"
else:
    print "Just right"
    
```

## Looping in Python – for loops

We can use a `for` loop to go through the same section of code more than once.

```
for i in range (10):  
    print "hello"  
print finished
```

```
for i in range (10):  
    print i*5  
print finished
```

Note that the loop starts with 0 and ends with 9.

Write your own loop to print the 6 times table. Your printout should look like this:

```
1x6= 6  
2x6= 12  
3x6= 18
```

and so on.

## Looping in Python – While loops

Sometimes we want to keep on doing something forever, or until something changes.

```
While it's raining, we'll play inside.  
While it's cheap, we'll buy some.
```

We can do this in Python too. One way to use this is to run a program over and over until the user enters something to stop it.

```
cont="y"  
while cont="y":  
    do whatever the program should do  
    cont=raw_input("Do you want to continue? y/n")  
print "Thank you for playing."
```

When the user enters anything other than "y" in response to "Do you want to continue?" the `while` loop stops because the test is no longer true.

Write a program that starts with 1 and doubles it and displays the results until the user no longer wants to continue.