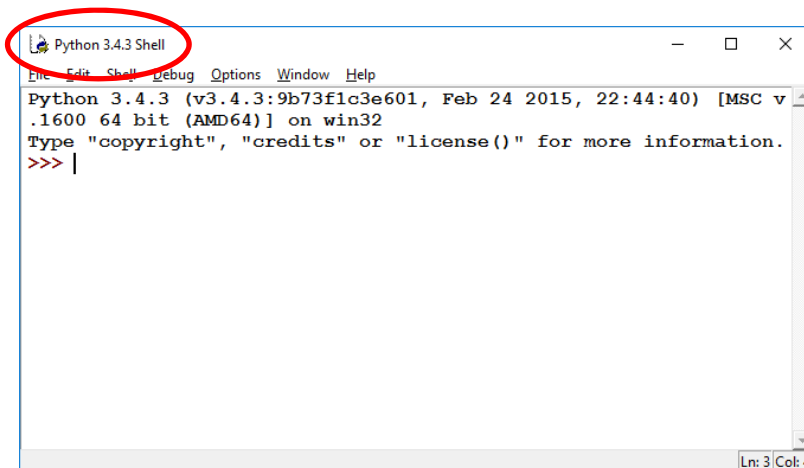


LAB 1: Introduction to writing computer programs

We all have used computers to do all sorts of useful and interesting things. Each application program in the computer responds in different ways to your input from the keyboard, mouse or a file. The response we get depends on the design of the application program we are using. In this course, you will learn to write your own computer programs, so you can give the computer instructions to react in the way *you* want.

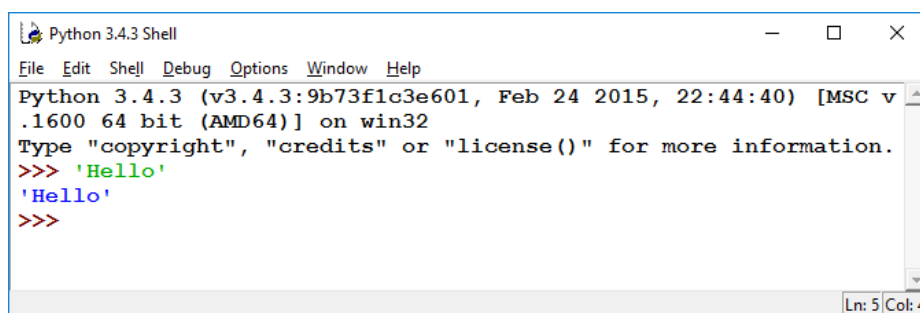
IDLE

IDLE (Integrated Development Environment) is a tool, normally called "the shell", which is an interactive environment. It is an easy way to check the syntax of the code instantly. It is a unique and very useful feature of Python.



IDLE consists of the command line prompt, an editor, and an interpreter. Everything typed after the command line prompt (>>>) are evaluated by the interpreter and the results are displayed instantly.

Type the following in python statement prompt:



The **string** is shown in green. The **output** is shown in blue.

The string is enclosed in a set of single quotes('), or a set of double quotes ("), or a set of triple quotes ("""). Try them

```
>>> 'Hello'
'Hello'
>>> "Hello"
'Hello'
>>> '''Hello'''
'Hello'
>>> |
```

If you wish to repeat a message a number of times, you can use the star (*) symbol.

```
>>> 'Hello '*3
'Hello Hello Hello '
>>> |
```

The interpreter takes the star key as repetition operation and duplicates the word 3 times. A useful situation for this is where, for example, you want to output 20 - characters:

```
>>> '_'*20
|_____|
>>> |
```

We could use the `print` function for more user-friendly output.

```
>>> print('_'*20)
|_____|
>>> |
```

TASK1: What difference can you see in the above two printed messages?

Python Scriptmode

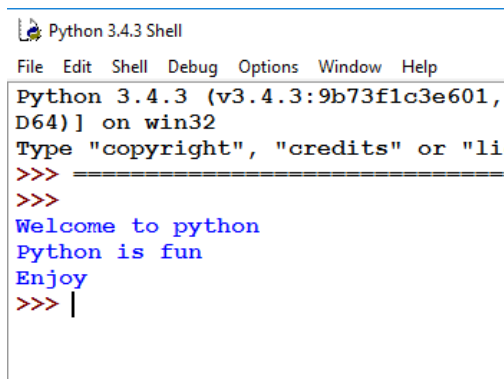
A limitation that comes with the python interactive mode is that, the statements you type on the python prompt are not saved – they are only available in the session (in memory). You can save the session text via **File – Save As**, but only as **txt file**, not executable python script.

To save the program for later use, create a text file. A text file can be created by clicking **File – New (or Ctrl + n)**. Name the file with an extension **.py** which is the *python script file*.

Example 1: Now type the following in a text file named **hello.py**

```
# Display two messages
print('Welcome to python')
print("Python is fun")
print(''''Enjoy''')
```

To run the script, **Run, Run module**. Or simply **press F5**. You should see the following



```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601,
D64)] on win32
Type "copyright", "credits" or "li
>>> =====
>>>
Welcome to python
Python is fun
Enjoy
>>> |
```

Explanation of the code

Line 1: **#Display two messages** is a comment statement.

Line 2-4: **print** statement prints the string on the monitor. The string can be started and ended using a set of single quotes (as in line 2) or a set of double quotes (as in line 3) or a set of triple quotes (as in line 4).

Note that the triple quote is made of three single quotes ☺

Comments

We normally use comments for two purposes:

1. To help whoever is reading the program understand how the program works. Comments are not programming statements and are ignored by the python interpreter.
2. To ignore a line of code out of the program instead of deleting it. If you delete a line and wish to use it later you will have to re-type it. By commenting out some code, it is easy to re-use it by removing the comment quotes.

Although comments are not part of the program logic - they are an important part of the program. Good programmers always have concise and meaningful comments to help others as well as themselves understand the program.

Line comments (#): When the interpreter sees the #, it ignores the rest of the statement. So the line comment can be used at the end of the command as well, as shown below

```
# Display two messages
print('Welcome to python') # used single quotes
print("Python is fun") # used double quotes
print(''Enjoy'') # used triple quotes
```

Paragraph comments ("""): When you want to write/block a set of lines. Use a set of *triple quotes*, as shown below

```
'''
Author: <your name>
Purpose: To get familiar with the output statements
'''

# Display two messages
print('Welcome to python') # used single quotes
print("Python is fun") # used double quotes
print(''Enjoy'') # used triple quotes
```

When the interpreter sees the triple quotes, it scans for the next triple quote and ignores any text between the two sets of triple quotes.

The print command

`print` is a function that outputs a piece of text you put between the brackets to the screen.

Note: python is case-sensitive. Which means the `print` is not the same as `Print` (capital letter P is not the same as lower letter p)

This is the same as languages like C#, or java. VB is case-insensitive.

TASK 2: To print `Let's do it` on the screen how would you write the print statement? Write any two of them below

Escape character

When you are required to use a quote (single or double) in your output and it clashes with the quote you use to show the string, like below

```
>>> print('Let's do it')
```

You get an error. Python thinks the string quote ends when it sees the second single quote. It considers it to be the end of the message and does not know what to do with the rest of the sentence.

To overcome this problem, we can use the escape character `\` as shown below

```
>>> print('Let\'s do it')
```

The quotation mark is a special character used to define the beginning and end of a string. By placing the `\` in front of it, the quotation mark is no longer considered a special character, i.e. it will be treated as a normal character (in that single instance only).

The escape character `\` can be used with `n` to draw a new line as shown below

`\n` places the text in a new line without having to use a separate `print` statement.

```
>>> print('Let\'s \ndo it')
Let's
do it
>>> |
```

TASK 3: Write two ways in which you can print the following
Hello good sir.
Are you having a good time?

Python Web and Multimedia

We can use python for web programming and multimedia files

```
>>> import webbrowser
>>> webbrowser.open("http://docs.python.org/3/")
True
>>> chrome = webbrowser.get("google-chrome")
>>> chrome.open("http://docs.python.org/")
True
>>> chrome.open("/home/alex/Videos/DM.mp4")
True
>>> firefox = webbrowser.get("firefox")
>>> firefox.open("/home/alex/Videos/DM.mp4")
True
```

Things to remember

Python is case sensitive.

Do not use the full-stop character after the statement (unless it is part of the string).

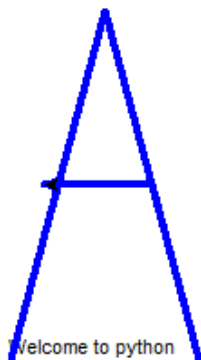
Turtle Graphics programming

Programming graphics is a great way to learn to write a computer program.

Example 2: Write the following to **draw the letter A**. You can first do this on the shell, where you can see the results immediately. Later on, as we get better with our programming skills, we can write it in a script.

```
>>> from turtle import *
>>> showturtle()
>>> title("Drawing with turtle")
>>> write("Welcome to python")
>>> pencolor("blue")
>>> pensize(4)
>>> left(75)
>>> fd(200)
>>> right(150)
>>> fd(200)
>>> back(100)
>>> right(100)
>>> undo()
>>> right(110)
>>> fd(30)
>>> undo()
>>> pos()
(77.65, 96.59)
>>> lt(5)
>>> fd(60)
>>>
```

You should see an output as shown below



TASK 4: Create a file called `letterA.py`, write the above as a script, and run it. Add comments to each line of code. Show it to your tutor when you have finished.

Note: `fd()` is the same as `forward()`.

TASK 5: Create a python script file called **square.py** and write in the following code.

```
'''
Author: <your name>
Purpose: To draw a square using turtle
'''

from turtle import *
forward(50)
right(90)
forward(50)
right(90)
forward(50)
right(90)
forward(50)
```

TASK 6: Create a file called **triangle.py** and write the script. Show it to your tutor when you've finished.

Turtle methods

Setting and measurement

- degrees()
- radians()

Pen control, drawing state

- pendown() pd() down()
- penup() pu() up()
- pensize() width()
- pen()
- isdown()

Color control

- color()
- pencolor()
- fillcolor()
- filling()
- begin_fill()
- end_fill()

More drawing control

- reset()
- clear()
- write()

Turtle state, visibility

showturtle() st()
hideturtle() ht()

Appearance

shape()

Using events

onclick()
onrelease()
ondrag()

Input methods

textinput()
numinput()

Methods specific to screen

bye()
exitonclick()
setup()
title()

More information on Turtle

- To find out more information/other commands in turtle, check out <http://docs.python.org/3/library>
- To get more information on a command, use the help. For example, to get more information on *pencolor* use `help(pencolor)`