Before entering the programming language...

```
Installation
    -win
    -linux
    -mac
how does it look like
    -terminal, console
what is needed to write code
    -text editor
    -tie the text editor to 'magic' command
paradigm of programing in Python aka style of code writing
    -dynamic
working process
    -edit-run-edit
```

Install

Python 3.2 vs python 2.7 It is important to clarify this difference \ http://inventwithpython.com/appendixa.html

```
       randn(4,5) in 2.7        ==  random.randn(4,5) otherwise it does not work
       print 'something' in 2.7 ==  print ('something') it is also a function

       installing python 3 on linux
       sudo apt-get install python3 python3.3-numpy python3.3-scipy python3-
   matplotlib python3-pandas

       To make Ipython -- pylab work, in addition one needs to instal:

       sudo apt-get install python3-tk
```

Ipython

```
       Simple installation of the program, and Ipython.
       For linux simple

       sudo apt-get install ipython3 ipython3-notebook

       Windows:
       python(x,y)
       http://code.google.com/p/pythonxy/wiki/Downloads

       Anaconda
       https://store.continuum.io/cshop/anaconda/
       and upgrade to python 3.3 as described here:
       http://www.walkingrandomly.com/?p=5089
```

```
You can have both versions 2.~ and 3.~ on the same machine. On linux it works very
nicely, with windows
ipython3 is, in my case, could not load matplotlib (Anaconda)(Python(x,y) is only
Py2.7).
```

To check anytime which version is on type in as a shell command (outside python env.) in terminal or console:

In [ ]:
```
python --version
```

```
inside python we can check the module's version
```

In [13]:
```
import numpy
```

In [14]:
```
numpy.version.version
```

Out[14]:
```
'1.7.1'
```

Text editor

```
Any that has :
    -syntax highligthing
    -can recognize file types and adjust the number of spaces to <TAB> key
        Python uses indentation to distinguish things and 4 spaces make one tab
    -can do area selection (on windows ALT+select) not just line select
    -comment and uncomment blocks of code with correct character for commenting '#' in
Python
    -a good option is autocompletion for commands
```

For Win I recommended Notepad++ http://notepad-plus-plus.org/download/v6.5.4.html

Or SciTe that comes with Python(x,y) distribution

How does it look like

Python can run in a console/terminal by typing in:

In [ ]:
```
python
```

to have more interesting command line look and capabilities use Ipython

In [ ]:
```
ipython --pylab
```

where the --pylab automatically loads Numpy,Scypi, Matplotlib libraries so we don't need to import

Also, in case the terminal style workflow is not appealing, one can just run the script file simply by by clicking on the file.

Working process

Interactive mode

In [ ]:
```
When loading 'python' you get the standard interactive window:
```

In [ ]:
```
balaaz@balaaz:~ > python
Python 2.7.5+ (default, Sep 19 2013, 13:48:49)
[GCC 4.8.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> x=range(0,10)
>>>
```

In [ ]:
```
We can call the 'ipython' and you get a bit different work environment
```

In [ ]:
```
balaaz@balaaz:~ > ipython --pylab
Python 2.7.5+ (default, Sep 19 2013, 13:48:49)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: TkAgg].
For more information, type 'help(pylab)'.

In [1]: print 'hello'
hello

In [2]: 2/3.
Out[2]: 0.6666666666666666

In [3]: _2+100
Out[3]: 100.66666666666667

In [4]:
```

```
In these cases you type line by line, and every input has a response.
In ipython you have them numbered, so you can call them by typing underscore and the
number of line
```

```
Another good feature of interactive ipython is the typing history. you can call
previous executions by
moving with up and down arrows on the keyboard
```

edit-run-edit mode

```
But to save large scripts the interactive sculpturing is not convenient. You write down
all in as a text,
give the text file '.py' extension and run it!
```

```
You can run it in shell level or from inside python.
```

to run from shell

In [ ]:
```
python script_name.py
```

to run from inside ipyton

In [ ]:
```
run script_name.py
```

```
For running the script inside python one needs to be in the directory where the script
is.
```

```
it is simple to navigate with 'cd' and 'cd ..' , these shell commands work inside
ipython same way
```

```
<TAB> completion :
    hitting the tabulator key after 'run' will list out all the scripts that you can
choose
    from the current directory, by starting to type a letter and then hit <TAB> will
narrow down the search
```

In [ ]:
```

```

When writing a script in the text editor, at every moment you can turn to a console to run it (or click on the file), so edit-run and then edit again if you want to add something to the code.

In [ ]:
```

```

In [ ]:
```
#//end of the 'before start'
```