Hands On Session 1

Floats, integers and func

1) Define a = 3.6, b = 2 and add them storing the result in c. What are the types of a, b and c?

In [1]:
```
a=3.6
b=2
c=a+b
print(type(a), type(b), type(c))
```

```
(<type 'float'>, <type 'int'>, <type 'float'>)
```

2) In a script, use an if statement to check if variable a is an integer

In [2]:
```
a = 5.0
b = 2
if (type(a) is int):
    print("a is an integer")
else:
    print("a is not an integer")
```

```
a is not an integer
```

3) Write a function to add two numbers and always return a float

In [3]:
```
def add(a, b):
    "Add a and b, always returns a float"
    return float(a) + float(b)

add(3,6)
```

Out[3]:
```
9.0
```

4) Define a function to raise a floating point number to an integer power N (Note power operator in Python to raise a to power N is a**N). What changes are needed for non-integer powers?

In [4]:
```
def power(a, N):
    return a**N
#Test fn
print(power(5.,2))
#No changes needed for non-integers power
#because of duck typing
print(power(9.,0.5))
```

```
25.0
3.0
```

More Advanced

1) Write a function which combines both 4) and 6) above to get the hypotenuse of a triangle from two side lenghts h2 = o2 + a2

```
In [5]: def pythag(o, a):
            return add(o**2,a**2)**0.5

        print("Try Pythagorean triple 3, 4, ", pythag(3., 4.))

        ('Try Pythagorean triple 3, 4, ', 5.0)
```

2) What does the add_fn do?

```
In [6]: def add_fn(a, b, fn):
            return fn(a) + fn(b)
```

The final argument is a function. We can pass the function itself in python to other functions (we only call it when we use brackets). In this example, fn is any function which accepts a single argument, so we can use it as follows:

```
In [7]: def add_fn(a, b, fn):
            return fn(a) + fn(b)

        #Convert two ints to float and add
        print(add_fn(3, 4, float))
        #Convert integers to strings and add together
        print(add_fn(3, 4, str))
        #Define square function and add squares of numnbers
        def square(a):
            return a**2
        print(add_fn(3, 4, square))

        7.0
        34
        25
```

3) Write a recursive factorial function (Note, if you don't know what recursion is, don't worry about this).

```
In [8]: def factorial( i ):
           if i <1:
               return 1
           else:
               out = i * factorial(i-1)
               return out

        factorial(4)

Out[8]: 24
```

Hands on session 2

String and files

1) Build a sentence 's' by defining and adding the 4 strings 'is' ,'a', 'this' and 'sentence' in the right order. Capitalise the first letter of each of the words. Print the first letter of each word. (note no unique way to do these).

```
In [9]: s = "this" + " " + "is" + " " + "a" + " " + "sentence"
        print(s.title())
        print([i[0] for i in s.split(" ")])

        This Is A Sentence
        ['t', 'i', 'a', 's']
```

2) Write a function to add a number and a name, e.g. 'filename0' from input 0 and 'filename' (note str(i) converts an int to a string)

In [10]:

```
def add_number(name, number):
    return name + str(number)

add_number("filename", 5)
```

Out[10]:

```
'filename5'
```

3) Download a text file (or create your own with notepad, must be plain text not word). Read the contents of the file into a string

In [12]:

```
#Code to create a file with some text
#(should be done outside of ipython but include here for completeness)
import os
os.system("echo 'some text for text file' > some_file")
#Code to create a file with some text

with open("some_file") as f:
    print(f.read())
```

```
some text for text file
```

4) Write a script to check if a keyword is inside a text file.

In [16]:

```
#Code to create a file with some text
#(should be done outside of ipython but include here for completeness)
import os
os.system("echo 'some text which includes \n keyword on a line \n in the tex
t file' > file.txt")
#Code to create a file with some text

with open('./file.txt') as f:
    for line in f.readlines():
        if "keyword" in line:
            print(line)
```

```
 keyword on a line
```

Hands on session 3

Lists and Iterators

1) Create a list with prime numbers 7,3,5,1,2 and sort it so they are in asending order. Use an iterator to loop through and print the output

In [19]:

```
l = [7,3,5,1,2]
l.sort()

for i in l:
    print(i)
```

```
1
2
3
5
7
```

2) Write a loop to print 10 strings with names: 'filename0', 'filename1' to 'filename9' (note str(i) converts an int to a string)

In [20]:

```
for i in range(10):
    print("filename" + str(i))
```

```
filename0
filename1
filename2
filename3
filename4
filename5
filename6
filename7
filename8
filename9
```

3) Using l = [1,2,3], write a loop to add a number to all elements giving [2,3,4]. Write a function to take in a list l and number N, which adds N to all elements of l and returns the list.

In [26]:

```
l = [1,2,3]
for i in range(len(l)):
    l[i] = l[i] + 1

print(l)

def add_to_list(l, N):
    for i in range(len(l)):
        l[i] = l[i] + N

    return l

print(add_to_list(l, 5))
```

```
[2, 3, 4]
[7, 8, 9]
```

More Advanced

Write a function which takes as an input the name of a file, reads it, removes all vowels (a, e, i, o and u) and returns a string made up of consonants

In [32]:

```
#Split the vowel removal into a seperate function so we can test it
def remove_vowels(filestr):
    #Could use replace vowel by vowel but
    #iterator is a nice solution
    for vowel in "aeiou":
        filestr = filestr.replace(vowel,"")
    return filestr

#Now read and use remove function
def read_file_and_remove_vowels(filename):
    with open(filename) as f:
        filestr = f.read()
    return remove_vowels(filestr)

print(remove_vowels("testing words"))

#Some file with "testing words in file"
import os
os.system("echo 'testing words in file' > file.txt")
print(read_file_and_remove_vowels("file.txt"))
```

```
tstng wrds
tstng wrds n fl
```

Hands on 4

Dictionaries, Numpy arrays and classes

1) Dictionary – Create a shape_sides dictionary d with keys "triangle", "square" and "pentagon" and values 3, 4 and 5 respectivly. Iterate and print all items

In [29]:

```
#Setup two elesments directly
d = {"triangle":3, "square":4}
#Add another
d["pentagon"] = 5
for key in d.keys():
    print(key, d[key])
```

```
('pentagon', 5)
('square', 4)
('triangle', 3)
```

2) Numpy arrays – Import the numpy module, create an numpy arrays of values from 1 to 5 and add one to each entry.

In [30]:

```
import numpy as np
x = np.array([1,2,3,4,5])
x = x + 1
print(x)
```

```
[2 3 4 5 6]
```

3) Classes – Create a class called number which takes an input x in its constructor and stores it (self.x = x). Add a method to square the (self.x) value and return

In [33]:

```
class number():
    #Constructor, takes in x and stores
    #is internal variable self.x
    def __init__(self, x):
        self.x = x
    def square(self):
        return self.x**2

n = number(4.)
n.square()
```

Out[33]:

```
16.0
```

4) Create a module containing a function which adds two numbers a and b, returning thier sum. import into a script and print output

In [34]:

```
#Add this function to a file called module
def add(a, b):
    return a + b

#Use system call to create module code
os.system("echo 'def add(a, b):\n    return a + b' > add_module.py")

#Import module and call add function
import add_module
output = add_module.add(3.,6.)
print(output)
```

```
9.0
```

More Advanced Questions

Read the input file to the right and store the results inside a dictionary with keys ”integer”, ”float” and ”string” and values stores with appropriate type

File: integer; 10 \n float; 1.25 \n string; “hello” \n

In [35]:

```
import os
#Create an input file out.txt for this example
os.system("echo 'integer; 10 \nfloat; 1.25 \nstring; hello' > out.txt")

# Empty dictonary
d = {}
with open("out.txt") as f:
    #Loop over each line in file
    for line in f:
        # split line using semi-colon into
        # key and value (plus remove end of line
        #character \n in case it's still there)
        key, value = line.replace("\n","").split(";")
        #Try to convert to integer
        try:
            d[key] = int(value)
        except ValueError:
            #If this fails, try to
            #convert to float
            try:
                d[key] = float(value)
            except ValueError:
                #Other save whatever it is
                d[key] = value

print(d)
```

```
{'integer': 10, 'float': 1.25, 'string': ' hello'}
```