Bioinformatics Programming Class 2

Aim of the exercise

In this class we will develop a simple calculator using python. We will create a script "calculate.py" that will work as follows:

```
./calculate.py plus 3 4
> 7
./calculate.py minus 8 2
> 6
```

But before we start this we will first introduce simple calculations in python.

The python shell

Type "python" in your terminal, to start the python shell. Type the following

```
>>> 3+4
```

```
>>> 8-2
```

That's nice isn't it? You can do some simple calculations straight away. The python shell is very good for testing some expressions.

Q1: Please do experiment, can you do some more advaced calculations? What about brackets?

Operators

What you have used above are numeric operators.

Type:

```
>>> 5/3
```

```
>>> 5.0 / 3.0
```

Q2: What is going on here? What is the difference between those operations (be quite exact in your description)?

Variables

Type:

```
>>> a=4
>>> b=5
>>> a

>>> b
```

Variables are the key to any programming language. In python you can assign a value to a variable with the "=" operator. Note that variables are quite like variables you may use in mathematical equations. However, there is a very important difference: in imperative programming languages like python the order of the assignments is crucial for the working of the program.

Type:

```
>>> a=3
>>> a=6
>>> b=5
>>> a + b
```

Q3: Now try and change the order of the assigments. What rule does python have about the order of the assignments?

Types

type:
```
>>> c = "Hello "
>>> d = "World"
>>> c + d
```

Q4: What does the "+" operator do in this context?

Q5: Try adding a and c. What does the error message mean?

Type conversions

type:
```
>>> str(4.0)
>>> int(4.0)
>>> int ("4")
>>> float(4)
```

Q6: Can you now add a+c, as in Q4 ?

Q7: Going back to question Q2, can you now explain why different answers where given?

Python Scripts

So far we have used the "python shell"; this is very useful for testing out code, but not so good for larger programs. We will now write a script in which we calculate the simple sums of the above exercises. Create a script "mySum.py" in an editor with a graphical user interface (e.g. gedit, nedit, x-emacs or kate) and run it ("./myScript.py" or "python myScript.py") – do not forget to use chmod to make the script executable.

```
#!/usr/bin/python
# The line above tells the system which python command it should use

# assign a and b
a=4
b=5

# calculate the result
answer = a+b

# print the result
print answer
```

Note that “answer” in the above example is just another variable, like “a” and “b”.

Q8: What type of arguments can you print with the print statement? - experiment in the python shell and also try a “,”.

Comments

In python comments start with a “#”. It is very important that you annotate your code with helpful comments. This is compulsory for any code that you will hand in during this project. Note that you should also choose sensible variable names, reflecting the kind of data they contain.

Q9: Be a little experimental, where in the line can the # be placed, and how does python interpret this? You can also experiment with this in the python shell.

Command Line Arguments

In the script above, we would have to change the code, if we wanted to calculate a different sum. This is not very helpful. We can give arguments to the script, that we can use in the program. Create a new script mySum.py that calculates the sum. For now you can use the following snippet of code to get arguments:

```
# import packages
import sys

# get arguments
argument1 = sys.argv[1]
argument2 = sys.argv[2]
```

Q10: What type does python assume the command line arguments have?

We would now like to extend our calculator further so that it can perform different calculations as mentioned in the start of the class.

if statement

The if statement is probably one of the most important concepts in programming, as you can get your program to make decisions.

Try the following

```
a=True
b=False
if(a):
        print “yes”
else:
        print “no”
```

Note the indentation is compulsory in python, the line after the if statement contains a tab. You can also use spaces – four spaces is quite common. Make it a habit for yourself to be consistent in this. Several editors (like emacs or kate) help you with this by doing automatic indentation if you press tab.

Q11: Which types are a and b here?

Comparison

Try:

```
a = “hello”
b= “hello”
c= “duck”
a==c
```

Q12: What kind of operator is “==” ? What type is its result?

Q13: What combination of types does it take?

Combining if statements of comparisons gives you a very powerful tool.

```
if(a>b):
        answer=True
elif(a<b):
        answer=False
```

```
else:
        answer= True
```

Q14: What does the above statement compute?

Calculator

You are now ready to combine all the pieces and write your calculator. Possible extensions:

- calculate integer and float division
- calculate absolute value
- checking if input is correct

Please do hand in your script in the next class: please hand it in on paper, so that we can give you some feedback.