**PYTHON**

Python is a widely used programming language and the most common programming language in bioinformatics. Key attributes are: (1) The language is accessible to read, to write, and to learn; (2) It is well-suited to object oriented programming (in brief, data structures and algorithms/ programming logic are joined rather than separate); (3) Because it is widely used, there are many available libraries and third party toolkits.

**Getting Python and Software to Write Code**

You should also install Python. I recommend using PyCharm as a text editor, but you can use the unix nano text editor.

**Your first program**

Your first program.

Let’s first write the program in nano. Create with a .py extension.

The print () function takes a list of arguments/ information and prints to your screen.

print ("Hello world\n")

Here there is one argument "Hello world." The quotation marks tell python that we are dealing with a string. You can use single or double quotes. You can put double quotes inside of single quotes and visa-versa. The \n is a newline character.

python helloworld.py

We are telling our computer to find the program python and send the contents of helloworld.py to this program

It is good to comment your text so people can follow your code.

#my first python program

print ("Hello world\n")

Python knows a hash in quotes is a hash

print ("This is a hash character #")

A large amount of a program can be “commented off” by *'''* at the start and the end of the commented text.

**Storing strings in variables**

Variables hold data, and the data they hold are allowed to vary. One type of variable in Python is a string that has a sequence of Unicode characters.

We can make a new variable and assign the variable a value. These names are case sensitive.

my\_dna=("AGCTAGCT")

print("my\_dna\t" + my\_dna)

Note that inside the quotes, text is reported. Outside the quotes, the variable value was extracted.

The print command can print multiple things if they are separated by +. This is referred to as concatenation.

Variables must begin with letters or an underscore character. After that, you can use letters/ numbers. It is advisable to use lower case.

examples:

dna=("ATGC") #you want to have a descriptive name when possible.

Use underscores for clarity.

first\_name = ("Harry")

last\_name = ("Potter")

#Note what is printed below. Is it clear why?

my\_dna=("AGCTAGCT")  
dna2='my\_dna'  
print ("1"+'dna2')  
print ("2"+ dna2)  
dna2=my\_dna  
print ("3"+ dna2)

**Tools for manipulating** **strings**

Often, we want to get the length of a string. We can use the len function.

x=len(my\_dna)

This works (though does not print).

However,

x=("length of my dna is\t"+len(my\_dna))

does not work because “TypeError: can only concatenate str (not "int") to str”

We get this message because *strings and numbers are different types of data!*

The str() function takes an argument, a non-string, and converts it into a string.

x=("length of my dna is\t"+str(len(my\_dna)))

print (x)

If we have a string that we need to convert into a number, int() works.

number= int("4")

print(number)

Other ways of manipulating string data use *methods*. A method is like a function that belongs to a particular data type. For example, the lower method belongs to the string type. When use a method, the variable comes first, followed by a period, then the name of the method, then the method arguments (if any) in parentheses. Of course this differs from a function like print() or len(). In this latter case, we write the function name first and the arguments go in parentheses.

print (my\_dna.lower())

As you may guess, you can’t use the *lower() method* on a non-string type. my\_number=len("AGCT")

print(my\_number.lower())

AttributeError: 'int' object has no attribute 'lower'

Try the method *upper*!

Another method, *replace*, replaces patterns elements in a string with something else.

variable=("bot")

variable.replace("b","c")

Note the original variable is not affected

variable.replace("bot","robot")

Still acts on the original variable. You need to assign the result to a new variable to capture the method’s output.

We can use brackets [.:.] to extract part of a string.

protein="mpadktnvd"

print(protein[3:5])

This command returns “dk” because the first value in the string is indexed as 0, and positions are inclusive at the beginning but exclusive at end. In other words, character 3 is included in the string and character 5 is not.

The .count("x") method; returns a number of the number of “x”s found in a string. Its single argument is the text you want to count.

print(protein.count("d"))

Finally, the .find("x") method; gives the location in the string (the index) of the pattern “x”

print(protein.find("d"))

Right now, we can only use these commands with exact substrings, but as you can imagine, they work with regular expressions. Note that if find does not identify the substring, a -1 is returned.

**Maths**

Of course, variables can hold both integers and decimals as well as strings. When Python encounters a mathematical statement e.g. x=3, it generates a variable named x and assigns the value 10 to the variable name.

x=3

y=2

sum=x + y + 3

print ("Sum of x and y is " + str(sum)+ "\n")

+, - , \*, /, %- modulo, remainder, x to the power of y is x \*\* y.

Note we are not assigning the results of each mathematical operation to any variable- instead putting output directly to the screen.

Other numeric functions:

print (abs(-x))  
print (int(3.14)) #does not round, just takes the integer.

The math module is provided by default by python, and it offers more functions such as log and factorial. We can import the module with the import command. A collection of useful modules, termed the “standard library” is bundled with python.

import math

print (math.factorial(x))

Python uses | as “or” in mathematical expressions.

Some operators have higher precedence than others, for example multiplication becomes before addition. A good strategy is to use parentheses to make operations clear regardless of the order of operation.

3+2\*5

Or (3+2)\*5

Mathematical shortcuts:

n=2

n= n +5

n += 5 #takes variable on the left and increments it by number on the right.

also -=; \*=, /=, ...

increment and decrement operators by 1

n+=1

n-=1