

## Science news of the day: “humanized” pigs!

### *In a First, Man Receives a Heart From a Genetically Altered Pig*

The breakthrough may lead one day to new supplies of animal organs for transplant into human patients.



Surgeons performed an eight-hour transplant of a genetically modified pig's heart at the University of Maryland School of Medicine on Friday. University of Maryland School of Medicine

<https://www.nytimes.com/2022/01/10/health/heart-transplant-pig-bennett.html>

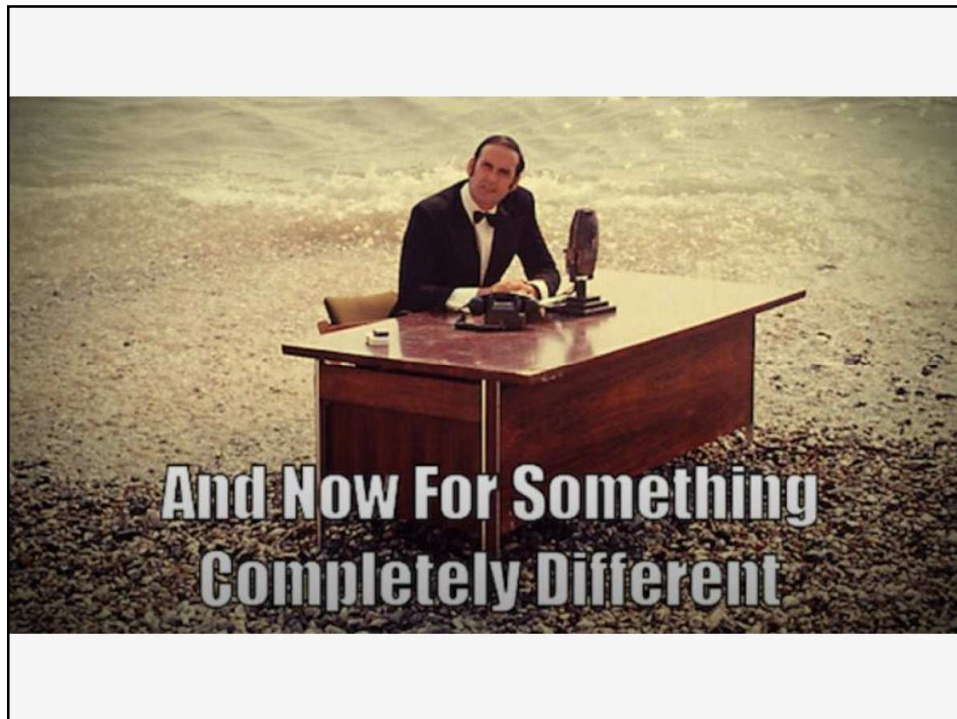
“The heart transplanted into Mr. Bennett came from a genetically altered pig provided by Revivicor, a regenerative medicine company based in Blacksburg, Va.”

“The pig had 10 genetic modifications. **Four genes were knocked out**, or inactivated, including one that encodes a molecule that causes an aggressive human rejection response.”

“A growth gene was ... inactivated to prevent the pig’s heart from continuing to grow...”

“In addition, **six human genes were inserted into the genome of the donor pig** — modifications designed to make the porcine organs more tolerable to the human immune system.”

1



2

## **A Python programming primer for biologists**

**(Named after *Monty Python's Flying Circus* & designed to be fun to use)**

**Systems Biology/Bioinformatics  
Edward Marcotte, Univ of Texas at Austin**

3

**In bioinformatics, you often want to do completely new analyses. Having the ability to program a computer opens all sorts of research opportunities. Plus, it's fun!**

**Most bioinformatics researchers use a scripting language, such as Python, Perl, or R, rather than a compiled language like C++**

**These languages are not the fastest, not the slowest, nor best, nor worst languages, but they're easy to learn and write, and for many reasons, are well-suited to bioinformatics.**

**We'll spend the next 2 lectures introducing Python to give you a sense for the language and help introduce the basics of algorithms.**

4

**Python documentation:** <http://www.python.org/doc/>  
**& tips:** <http://www.tutorialspoint.com/python>

**Good introductory Python books:**

- *Learning Python*, Mark Lutz & David Ascher, O'Reilly Media
- *Bioinformatics Programming Using Python: Practical Programming for Biological Data*, Mitchell Model, O'Reilly

**Good intro video (from a 2 day intro class at Google):**

- <https://www.youtube.com/playlist?list=PLC8825D0450647509>

**Practical Python, a self-paced online intro course:**

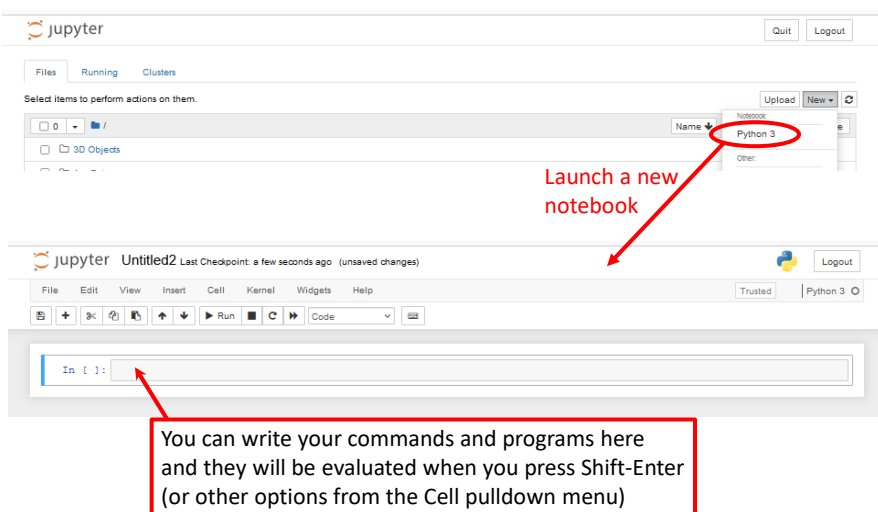
- <https://dabeaz-course.github.io/practical-python/>

**An online Python tutor with a nice interactive code viewer:**

- <http://www.pythontutor.com/>

5

**By now, you should have installed Python on your computer.  
If you're using Anaconda/Jupyter, it runs in a web browser:**

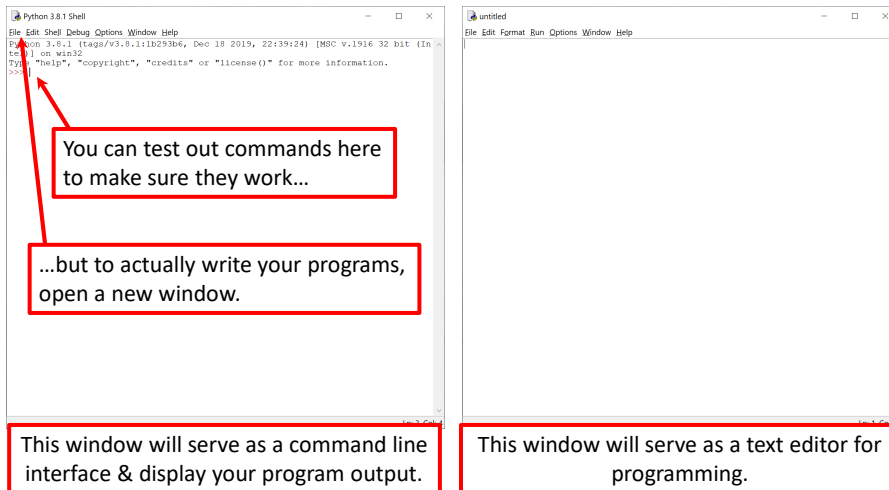


The image shows two screenshots of the Jupyter web interface. The top screenshot displays a file browser with a 'New' button and a dropdown menu where 'Python 3' is selected. A red arrow points from the text 'Launch a new notebook' to the 'Python 3' option. The bottom screenshot shows a code editor with a text input field containing 'In [ ]:'. A red box highlights this field with the text 'You can write your commands and programs here and they will be evaluated when you press Shift-Enter (or other options from the Cell pulldown menu)'.

6

## Or if you installed IDLE by following the instructions in Rosalind Homework problem #1:

Launch IDLE:



7

**Let's start with some simple programs in Python:**

**A very simple example is:**

```
print("Hello, future bioinformatician!") # print out the greeting
```

**Run the program. In Jupyter, you can just type Shift-Enter & the output will appear below this cell of the notebook.**

**The output looks like this:**

```
Hello, future bioinformatician!
```

FYI: This is version agnostic. Python 3 takes `print("X")`. Python 2 also takes `print "X"` as in Rosalind

8

### A slightly more sophisticated version:

```
name = input("What is your name? ")      # asks a question and saves the answer
                                         # in the variable "name"
print("Hello, future bioinformatician " + name + "!") # print out the greeting
```

### When you run it this time, the output looks like:

What is your name?

### If you type in your name, followed by the enter key, the program will print:

Hello, future bioinformatician Alice!

FYI: Python 2.x uses `raw_input()` instead of `input()`

9


## GENERAL CONCEPTS

Names, numbers, words, etc. are stored as *variables*.

Variables in Python can be named essentially anything except words Python uses as command.

For example:

```
BobsSocialSecurityNumber = 456249685
mole = 6.022e-23
password = "7 infinite fields of blue"
```

 Note that strings of letters and/or numbers are in quotes, unlike numerical values.

10

## ***LISTS***

Groups of variables can be stored as lists.

A list is a numbered series of values,  
like a vector, an array, or a matrix.

Lists are variables, so you can name them just as you would name any other variable.

Individual elements of the list can be referred to using [] notation:

```
The list nucleotides might contain the elements
nucleotides[0] = "A"
nucleotides[1] = "C"
nucleotides[2] = "G"
nucleotides[3] = "T"
```

(Notice the numbering starts from zero. This is standard in Python.)

11

## ***DICTIONARIES***

A VERY useful variation on lists is called a ***dictionary*** or *dict* (sometimes also called a *hash*).

→ Groups of values indexed not with numbers (although they could be) but with other values.

Individual hash elements are accessed like array elements:

For example, we could store the genetic code in a hash named *codons*, which might contain 64 entries, one for each codon, e.g.

```
codons["ATG"] = "Methionine"
codons["TAG"] = "Stop codon"
etc...
```

12

**Now, for some control over what happens in programs.**

There are two very important ways to control the logical flow of your programs:

**if statements**

and

**for loops**

There are some other ways too, but this will get you going for now.

13

***if statements***

```
if dnaTriplet == "ATG":  
    # Start translating here. We're not going to write this part  
    # since we're really just learning about IF statements  
else:  
    # Read another codon
```

**Python cares about the white space (tabs & spaces) you use!**  
This is how it knows where the conditional actions that follow begin and end. **These conditional steps must *always* be indented by the same number of spaces (e.g., 4).**

I recommend using a tab (rather than spaces) so you're always consistent.

14

Note: in the sense of performing a comparison, not as in setting a value.

== equals  
!= is not equal to  
< is less than  
> is greater than  
<= is less than or equal to  
>= is greater than or equal to

**Can nest these using parentheses and Boolean operations, such as *and*, *not*, or *or*, e.g.:**

```
if dnaTriplet == "TAA" or dnaTriplet == "TAG" or dnaTriplet == "TGA":  
    print("Reached stop codon")
```

15

### ***for loops***

Often, we'd like to perform the same command repeatedly or with slight variations.

For example, to calculate the mean value of the number in an array, we might try:

- Take each value in the array in turn.
- Add each value to a running sum.
- Divide the total by the number of values.

16



**In Python, you could write this as:**

```
grades = [93, 95, 87, 63, 75] # create a list of grades
sum = 0.0 # variable to store the sum

for grade in grades: # iterate over the list
    sum = sum + grade # add the grade to the sum

mean = sum / 5 # now calculate the average grade

print ("The average grade is ",mean) # print the results
```

In general, Python cares whether numbers are **integers** or **floating point** (also **long integers** and **complex numbers**).  
You can tell Python you want floating point by defining your variables accordingly (e.g.,  $X = 1.0$  versus  $X = 1$ )

Python 2	Python 3
>>> 2 / 3 0	>>> 2 / 3 0.666666

Python 2.x: print ("The average grade is "),mean

17

In general, Python will perform most mathematical operations, e.g.

**multiplication**      ( $A * B$ )  
**division**            ( $A / B$ )  
**exponentiation**    ( $A ** B$ )  
etc.

There are lots of advanced mathematical capabilities you can explore later on.

18

## READING FILES

You can use a *for* loop to read text files line by line:

```
count = 0
file = open("mygenomefile", "r")
for raw_line in file:
    line = raw_line.rstrip("\r\n")
    words = line.split(" ")

    # Print the appropriate word:
    print ("The first word of line {0} of the file is {1}".format(count, words[0]))
    count += 1

file.close()
print ("Read in {0} lines\n".format(count))
```

Stands for "read"

\r = carriage return  
\n = newline

Increment counter by 1

Placeholders (e.g., {0}) in the print statement indicate variables listed at the end of the line after the format command

Note: Python expects the file to be in your working directory or that you give it a full path.

19

## WRITING FILES

Same as reading files, but use "w" for 'write':

```
file = open("test_file", "w")
file.write("Hello!\n")
file.write("Goodbye!\n")
file.close()

# close the file as you did before
```

Unless you specify otherwise, you can find the new text file you created (test\_file) in the default Python directory on your computer. In Jupyter, you should see now it appear in the Jupyter home page directory.

20

## PUTTING IT ALL TOGETHER

```
seq_filename = "Ecoli_genome.txt"
total_length = 0
nucleotide = {} # create an empty dictionary

seq_file = open(seq_filename, "r")
for raw_line in seq_file:
    line = raw_line.rstrip("\r\n")
    length = len(line) # Python function to calculate the length of a string
    for nuc in line:
        if nuc not in nucleotide:
            nucleotide[nuc] = 1
        else:
            nucleotide[nuc] += 1
    total_length += length

seq_file.close()

for n in nucleotide.keys():
    fraction = 100.0 * nucleotide[n] / total_length
    print ("The nucleotide {0} occurs {1} times, or {2} %".format(n, nucleotide[n], fraction))
```

21

Let's choose the input DNA sequence in the file to be the genome of *E. coli*, available the class web site (& originally from the **Entrez genomes** web site)

The format of the file is ~77,000 lines of A's, C's, G's and T's:  
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAGAGTGTG  
TGATAGCAGCTTCTGAAGTGGTTACCTGCCGTGAGTAAATTTAAATTTATTGACTTAGG  
TCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTAC  
ACAACATCCATGAAACGCATTAGCACCACCATTACCACCACCATCACCATTACCACAGGT  
etc...

### Running the program produces the output:

The nucleotide A occurs 1142136 times, or 24.619133255346103 %  
The nucleotide G occurs 1176775 times, or 25.365788782211496 %  
The nucleotide C occurs 1179433 times, or 25.42308288395832 %  
The nucleotide T occurs 1140877 times, or 24.591995078484082 %

So, now we know that the four nucleotides are present in roughly equal numbers in the *E. coli* genome.

22