

---

## Introduction to Programming with Python

---

### A Useful Reference

<http://www.pasteur.fr/formation/infobio/python/>

## What is Computer Programming?

- An **algorithm** is a series of steps for solving a problem
- A **programming language** is a way to express our algorithm to a computer
- **Programming** is the process of writing instructions (i.e., an algorithm) to a computer for the purpose of solving a problem

We will be using the programming language **Python**

## Variables

- A **variable** is a mnemonic name for something that may change value over time.

```
kozak = "ACCATGG"
name = "Brian"
year = 2007
year = 2008
GC_content = 0.46
variable_name = value (generic variable assignment)
```

- Variable - "I don't think that word means what you think it means"

```
2008 = year (wrong!)
0.46 = GC_content (wrong!)
```

## Types

- Variables store values of some **type**. Types have **operators** associated with them.

```
year = 2008
nextYear = year + 1
GC_content = 2.0 * 0.21
kozak = "ACC" + "ACCATGG"
year = year + 1
kozak = kozak + "TT" + kozak
variable_name = value (generic variable assignment)
```

- You can have the computer tell you the value of a variable

```
print nextYear
print "The GC content is:", GC_content
print year
print kozak
```

## Strings

- Strings are a sequence of characters

```
kozak = "ACCATGG"
```

- Strings are index-able

```
kozak[0] refers to 'A', the first character in kozak
kozak[4] refers to 'T', the fifth character in kozak
```

- Strings have lots of operations

```
kozak.lower() returns "accatgg"
kozak.count('A') returns 2
kozak.replace('A', 'q') returns "qCCqTGG"
len(kozak) returns 7
```

## Nucleotide Content

```
kozak = "ACCATGG"
```

- What percent of the sequence corresponds to adenine nucleotides?

```
numberOfAdenines = kozak.count('A')
totalNucleotides = len(kozak)
A_content = numberOfAdenines / totalNucleotides
print A_content
```

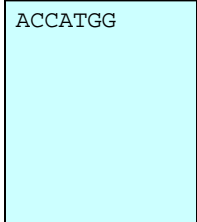
What went wrong?

## Reading a File

- Suppose the DNA sequence is stored in the file kozak.txt. We can read the sequence from the file...

```
file = open("kozak.txt")
sequence = file.read()
print sequence
```

kozak.txt



ACCATGG

Generic code for reading a file

```
variable_name_1 = open(string referring to file name)
```

```
variable_name_2 = variable_name_1.read()
```

## Putting it all Together

```
# Read in file and store string in variable *sequence*
file = open("kozak.txt")
sequence = file.read()
```

kozak.txt

ACCATGG

```
# Calculate number of adenines in sequence
numberOfAdenines = float(sequence.count('A'))
totalNucleotides = float(len(sequence))
A_content = numberOfAdenines / totalNucleotides
print A_content
```

What about GC content?

## Slicing a String

```
# Read in file and store string in variable *sequence*
file = open("kozak.txt")
sequence = file.read()
```

kozak.txt

ACCATGG

```
# Grab a piece of the sequence
firstThreeLetters = sequence[0:3]
print firstThreeLetters
middleThreeLetters = sequence[2:5]
print middleThreeLetters
```

What about *your* gene?

# Booleans

```
kozak = "ACCATGG"
```

- Booleans are either True or False

```
kozak == "ACCATGG"  
kozak == "GCATCAG"  
kozak == "accatgg"  
kozak.lower() == "accatgg"  
len(kozak) > 10  
len(kozak) < 10  
'A' in kozak  
'U' in kozak
```

# Decisions, Decisions, Decisions

- Normal execution flow: SEQUENTIAL
- Often you want to execute code (instructions) only in certain circumstances (i.e., conditionally)

```
file = open("kozak.txt")  
sequence = file.read()  
  
# Do we have a short sequence?  
if (len(sequence) < 50):  
    print "This is a short sequence."  
  
# Is this an RNA sequence?  
if (sequence.count('U') > 0):  
    print "Sequence has RNA nucleotides"  
  
# Check if sequence starts off looking like a gene  
if (sequence[0:3] == "ATG"):  
    print "Sequence has start codon."  
    length = len(sequence)  
    finalCodon = sequence[length-3:length]  
    print "Final three NTs are: " + finalCodon
```

## Otherwise...

- Sometimes you want to decide between two alternatives

```
file = open("kozak.txt")
sequence = file.read()

# Do we have a short sequence?
if (len(sequence) < 50):
    print "This is a short sequence."
else:
    print "This is a long sequence."

# Is this an RNA sequence?
if (sequence.count('U') > 0):
    print "Sequence has RNA nucleotides"
else:
    numofThymines = sequence.count('T')
    print "Sequence has ", numofThymines, " thymines."
```

## Nesting Conditionals

- You can put any code in the body of a conditional statement, including other conditional statements

```
# Check if sequence starts and ends looking like a gene
if (sequence[0:3] == "ATG"):
    print "Sequence has start codon."
    length = len(sequence)
    finalCodon = sequence[length-3:length]
    if (finalCodon == "TGA"):
        print "Sequence has stop codon."
    if (finalCodon == "TAG"):
        print "Sequence has stop codon."
    if (finalCodon == "TAA"):
        print "Sequence has stop codon."

# Is this an RNA sequence?
if (sequence.count('U') > 0):
    print "Sequence has RNA nucleotides"
else:
    if (sequence.count('T') > 0):
        print "Sequence has DNA nucleotides."
```

## Generic Conditionals

**# if-then**

if (boolean\_expression):

**# Statements to execute if boolean\_expression is true**

**# if-then-else**

if (boolean\_expression):

**# Statements to execute if boolean\_expression is true**

else:

**# Statements to execute if boolean\_expression is false**

**# nested conditionals**

if (boolean\_expression\_1):

    if (boolean\_expression\_2):

**# Statements to execute if boolean\_expression\_2 is true**

    else:

**# Statements to execute if boolean\_expression\_2 is false**

## Reading in a FASTA File

## Repetition is a Powerful Idea

- Suppose you want to repeat a series of instructions

```
# Tell us how you feel about this class
counter = 5
while (counter > 0):
    print "I love Bioinformatics!"
    counter = counter - 1

# Assuming we have a coding sequence, print out each codon
startOfCodon = 0
while (startOfCodon < len(sequence)):
    codon = sequence[startOfCodon:startOfCodon+3]
    print codon
    startOfCodon = startOfCodon + 3
```

## Loop (i.e., Repetition) Examples

```
# Find the start of all possible ORFs in sequence
startOfCodon = 0
while (startOfCodon < len(sequence)):
    codon = sequence[startOfCodon:startOfCodon+3]
    if (codon == "ATG"):
        print "Found start codon at ", startOfCodon
    startOfCodon = startOfCodon + 1

# Search for ambiguous nucleotides in sequence
indexOfCurrentNucleotide = 0
while (indexOfCurrentNucleotide < len(sequence)):
    if (sequence[indexOfCurrentNucleotide] not in "ACGT"):
        print "I don't recognize the character: ",
            sequence[indexOfCurrentNucleotide]
    indexOfCurrentNucleotide = indexOfCurrentNucleotide + 1
```

## Generic Repetition

```
# Loop
while (boolean_expression):
    # Statements to execute as long as boolean_expression is true.
    # Statements should ensure that, eventually, boolean_expression
    # will be false. Otherwise, the loop will repeat indefinitely.
```

## Python Summary

- Types of variables: numbers, strings, Booleans
- Assigning values to variables
- Slicing and dicing with strings
- Reading in files; text and variable value output
- Conditionals (if-then, if-then-else)
- Repetition Repetition Repetition Repetition  
Repetition Repetition Repetition Repetition  
Repetition Repetition Repetition Repetition...