Fun Python

Instructor: Rada Mihalcea

Notes from a class held online between April - May 2020 with students age 10-12

Lesson 0: Setting things up. Hello world!

What we will learn in this class:

- How to set up Google Colab to start programming "in the cloud"
- Write our first "Hello world!" program

Getting Ready

What you need:

- A computer connected to the Internet
- A browser
- A gmail account

First, we will create a place on the computer where we will work on our programs.

1. Open a browser.
2. Go to drive.google.com. Login into your Google account

3. On the left side of the browser, go to New / Folder. Create a new folder <*YourName*>*FunCode*. This will be your place to keep all your programs for this class.
4. Share this folder with your teacher: in the upper right corner, click on Share. Then when asked for an email address, enter <email-address-removed>.
5. Double click to go inside the folder <*YourName*>*FunCode*

Next, we want to make sure we have Google Colaboratory ready to go. Google Colaboratory is a program made available by Google, who lets us work on programs "in the cloud." That means we will write and run our programs on Google's computers. How cool is that! Can you imagine that? Your awesome programs sitting right next to other cool programs, such as the ones that let you search on Google?

6. Go to New / More / + Connect more apps. Search for Colaboratory. Click Install
7. You should be ready to write your first program using Colab!

Your First Program: Hello World

Let's write our first program, and have it tell the world "Hello world!" :)

Note: "Hello world" has been so often used as an example by programming teachers, that it now symbolizes the time when somebody enters the programming world. Like you, now!

1. Open a browser, go to drive.google.com, login into your Google account, go into <*YourName*>*FunCode* (remember you need to double click to go into a folder)
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to HelloWorld. Keep the ipynb extension -- this says this is a Python notebook, your first Python program!
3. Now we will write some code. In the first "cell", write
   print("Hello World")
   Now click on the right-arrow on the left side. This will run your first program, and you will see "Hello World" displayed.
4. Try printing another message. Did it work?

Lesson 1: Variables. Math with big numbers.

What we will learn in this class:

- Learn about variables
- Learn about input/output
- Learn about errors
- Putting it all together: A program that multiplies two numbers given by the user
- Assignment 1: Write a program that helps the user do some math operations

Variables

In real life, we use names to refer to the people and things around us. For instance, my name is "Rada" - so you know when you say Rada, it refers to your Python teacher, who aside from teaching Python does a lot of other things: programs, reads, bikes, …. Every time you want to refer to Rada, you do not need to describe who Rada is, but just use the name.

Variables in computer programs have the same roles: they are names we give to things, so we do not have to re-describe them every time we use them. And the fun thing is we can call them *anything* we want! In fact, computer programs can be quite fun to read because of all these names. So let's try some variables!

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to MathWithPython. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Now write

```
number1 = 5
number2 = 3
print("The product of the two numbers is ",number1*number2)
```

   3.1. Try running this program. Is the result correct?

   Now try changing the numbers to something much larger (e.g., Zara tried a number with 20 different digits. Can you beat that? :). Run again…

   3.2 Try changing the program so it adds the numbers.

Reading from the user

One way to set the value of a variable is inside the program, as we have done earlier. The other way is to ask the user of your program to tell you a value.

4. While you are in your MathWithPython program, go to the upper left corner, and click on +Code. This will let you create a new code piece.
5. Now let's ask the user to give us a number.

```
number1 = input("Enter a number")
number2 = input("Now enter another number")
print("The product of the two numbers is",number1*number2)
```

   Is the result correct? Why or why not?

Errors

When you ran the previous program, you most likely got an error. This is absolutely normal. Even now, after 30 years of doing all sorts of programming, I still get errors. When we get an error, we read it to understand what's wrong, and try to fix it.

Earlier, the error likely said "`TypeError: can't multiply sequence by non-int of type 'str'`" This is because when we read something from the user, what we get back is what is called a "string" - something like "Rada" or "today". We cannot multiply Rada or today, can we? We have to tell the computer that what we got back from the user are numbers! We will do this using the function "int" - see below.

```
number1 = input("Enter a number")
number2 = input("Now enter another number")
print("The product of the two numbers is",int(number1)*int(number2))
```

Does it work? It should -- and that is because we told the computer "look, these are numbers, you need to deal with them as numbers, not as names"

Extra: If you want to try something else, change the program above so that you do the sum of the numbers (and make sure you change your message too)

Assignment.

Create a new piece of code inside your MathWithPython program (remember you go in the upper left corner, and click on +Code). Write a program that does the following:

- Greets the user (with something like "Hello", "Hi", "What's up")
- Tells the user you will help them do some math
- Ask the users for two numbers
- Prints the sum, product, subtraction, and division of the two numbers. (consider making it clear, by telling the user e.g., "The product is")

And of course, feel free to try programming other things. The more you practice, the better you will get!

Lesson 2: If-then-else

What we will learn in this class:

- Quick review: input/output, variables
- Learn about formatting your code
- Learn about conditionals: if-then-else
- Learn about libraries: the random library
- Putting it all together: compare two numbers
- Assignment 2: Write our first game: a number game!

Review program input/output.
How do we “send” something to the output? In other words, what function do we use to write something for the user?
How about reading from the input? In other words, what function do we use to “capture” what the user is telling us?
Why do we use quotes in any message we include in these functions?
Bonus question: when we write a program, who are we? In the imagine below -- where are “we” as programmers?

Review variables.
What is a variable?
Why do we need variables?
When we read a variable from the user, what type is that variable? A number? A string?
How do we convert a variable to make it an integer, so we can do math operations?
Bonus question: How do we convert a variable to make it a real number, so we can do math operations with decimals?

Formatting your code.
Before we proceed to other interesting things we can do with a computer, it’s important to note a few things about writing code. We will come back to this as we move along in our class.

First, we need to pay attention to how we write and spell things. Computers are very … exact, they always want to hear the same thing they are used to. If we make a tiny change, they don’t like that. For instance, try changing one of your previous programs to use `Print` instead of `print`. Does it work? Most likely not. Or, try not including the quotes in `print("Hello world")`, and it will not work anymore. Luckily, for most of these mistakes, whenever we make them (yes, it does happen all the time that we forget a quote), the computer will give us an error message so we know we have to fix it.

Second, in this class, we will start seeing that even the spaces we use can make a difference. Python uses indentation to indicate that a certain block of code goes together. Here is a piece of code that tells us that the two print statements should be executed together ‘as a block’.

```
if(5 > 1):
    print("Hi")
    print("What's up")
```

If-then-else.
So far, everything we wrote in our program was executed when we ran the program. (that is, the computer "ran" each command, one by one). But what if we want to execute a command only in certain situations? For instance, for your first assignment, I asked you to do the division of two numbers. What if the divisor was 0? Or, we could write a program to talk to a user (we will do that soon!), but we want to say certain things only if the user said "*Yes*" To do this, we use an "if-then-else" command.

if-then-else syntax

```
if(condition):
  <do-this-if-condition-true>
else:
  <do-this-if-condition-false>
```

condition: is something that is either true or false (for instance, a comparison)
<do-this-if-condition-true>: is one or more commands that are executed when the condition is true
<do-this-if-condition-false>: is one or more commands that are executed when the condition is false

What will this piece of code do?

```
if(5 < 1):
   print("It's sunny today")
else:
   print("It's rainy today")
```

What will this piece of code do?

```
x = 10
if( x / 2 > 6):
   print("It's sunny today")
else:
   print("It's rainy today")
```

What will this piece of code do?

```
x = 10
if( x / 2 > 6):
   print("It's sunny today")
```

Important! Notice the indentation (empty spaces) before print. This tells the computer the print statements are to be executed only when the condition evaluates to true. (the same goes for "else": it requires a block of code that has indentation)

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to NumberGame. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Now let's ask the user for two numbers as we have done before, and tell the user which number is bigger.

   ```
   x = input("Enter a number")
   y = input("Now enter another number")
   if(int(x) > int(y)):
      print("The first number is larger")
   else:
      print("The second number is larger")
   ```

   Question: why did we use int?
4. Try running your code. Try running again with other numbers. What happens if the two numbers are equal?

Libraries.
Python has a LOT of commands *(or functions - but we'll talk about this later)* that do interesting things that are already implemented. That is, somebody else took the time to write them, and now they are available for us to use. This is a very cool thing, because it means we can be much more efficient in what we write, and we don't have to re-write all those commands that others have already spent time writing. The same will happen with code that we write, we can make it available for others to use!

These commands are organized into libraries -- yes, pretty much like the books in a library. If we want to have access to one of these libraries, so we can use the commands inside, we will have to tell the computer we want access. We do that with the command `import`.

Let's try using a library that is one of my favorites, because it lets us create random things. How fun!

1. While you are in your NumberGame program, go to the upper left corner, and click on +Code. This will let you create a new code piece.
2. Now let's tell the computer we want to have access to the random library, and that we want to create a random number. We do this with a command that is available in the random library: While you are in your HelloWorld program, go to the upper left corner, and click on +Code: `random.randint(a,b)`
   This will write a random integer that is larger or equal to a, and smaller or equal to b.
3. Try this code

   ```
   import random
   print(random.randint(1,7))
   ```

   When you run the code, what do you get? Do you all get the same number?
   Try now changing the code so that you get a random integer between 50 and 100.

Assignment. Putting it all together: A number game!

Let's now try to put together what we learn today. As an assignment for next time, you will write a number game.

1. Please write it inside the NumberGame program you created earlier. Go to the upper left corner, and click on +Code. This will let you create a new code piece.
2. Here is what the program should do:
   - The computer "thinks" of a random number between 1 and 100
     Hint:

     ```
     import random
     computerNumber = random.randint(1,7))
     ```

   - The program asks the user to say a random number
   - The program compares the two numbers, and if the computer number is larger, it says "I won", if the user number is larger, it says "You won"
   - Play the game three times. How many times did the user win?
3. Bonus: Add an explanation. That is, write a message in which you tell the user the computer number and the user number (something like "My number was …, your number was …"

Lesson 3: Lists, Fun text generator

What we will learn in this class:

- Go over the first assignment: (one possible) solution and improvements (as suggested by all of you!)
- Quick review: if-then-else, libraries
- Learn about lists
- Putting it all together: generate some funny sentences
- Assignment 3: Write a program that does MadLib games

Review of the first assignment.
Remember the assignment guidelines:
Write a program that does the following:

- *Greets the user (with something like "Hello", "Hi", "What's up")*
- *Tells the user you will help them do some math*
- *Ask the users for two numbers*
- *Prints the sum, product, subtraction, and division of the two numbers. (consider making it clear, by telling the user e.g., "The product is")*

```
print ("Hi! I will help you with some math!")
a = input ("Tell me a number.")
b = input ("Tell me another number:")
print("This is the addition of those two numbers: ", int(a)+int(b))
print("This is the multiplication of those two numbers: ",int(a)*int(b))
print("This is the division of those two numbers: ", int(a)/int(b))
print("This is the subtraction of those two numbers: ", int(a)-int(b))
```

Quick review: If-then-else
Why do we need if-then-else?
What comes right after the keyword "if"?

Is this correct? Why or why not?

```
if(7>5):
 print("Wow!")
else:
 print("Hmm")
```

What will be the output of this code?

```
if(7>5):
 print("Wow!")
else:
 print()

print("Hmm")
```

Quick review: Libraries.
Python has a LOT of commands *(or functions - but we'll talk about this later)* that do interesting things that are already implemented. That is, somebody else took the time to write them, and now they are available for us to use. This is a very cool thing, because it means we can be much more efficient in what we write, and we don't have to re-write all those commands that others have already spent time writing. The same will happen with code that we write, we can make it available for others to use!

```
import random
print (random.randint(1,7))
print (random.random())
```

Trivia: where does the name Python come from?

Lists.

So far, we have worked with several variables, of different *types* -- like integers, or strings. But we only worked with one variable at a time. What if we want to have a collection of variables? For instance, what if we needed 4 numbers? We could do something like:

```
n1 = 1
n2 = 5
n3 = 8
n4 = 3
```

Which is all fine, except that it's not that practical to keep naming variables, in particular when they seem to serve the same purpose (like here, we have four different numbers), and even more so when I have a lot of such variables. For example, what if I wanted to have 10 numbers? Fortunately Python (and many other programming languages) have lists!

A list is a collection of items. It is very commonly used in Python, and it can be used to store numbers, strings, a mix of numbers and strings, and even other lists!

A list always consists of zero or more items, between square brackets.

Here are some examples:

```
myList = []
myList = [1,5,2]
myList = ["one", "two", "seven"]
myList = [0,"Rada",4.2]
myList = [[0,1],"hello"]
```

What can we do with lists? Loooots of things! I don't even know them all :) (As it turns out, we often have to search up for how to do things in programming, and we often discover new things we didn't know!)

```
myList = [3,245,4]
print(myList)
print(myList[0])
print(myList[2])
myList.sort()
print(myList)
myList.reverse()
print(myList)
myList.append(25)
print(myList)
```

```
myList.extend([3,4])
print(myList)
print(myList[0:2])
```

We can even get a random element from a list!

```
import random
nouns = ["apple", "pear", "sardines"]
print(random.choice(nouns))
```

Important! Operations on lists are done *in place*, which means that they are applied directly on the list. For instance, if I do

```
myList = [1,3,9,2,0]
myList.sort()
```

myList will be now `[0,1,2,3,9]`
(and I don't have my older list `[1,3,9,2,0]`)

Putting it all together: Let's generate some fun text
We will use the list data type we just learned about to create some lists of words, and then randomly choose from those lists to generate a sentence.

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to CoolLists. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Let's create a list of words.
   ```
   nouns = ["sun", "dog", "water", "fish", "dish"]
   ```
4. Try printing the first item. How about the third item?
5. Try sorting the list.
6. Now let's create more lists of words. Add a list of adjectives, and a list of verbs. I suggest you include 5 words in each list.
   Hint: use only singular nouns; use only verbs third person singular that can have a direct object
7. Now let's try writing a sentence. , using random words selected from your lists.
   For instance, I can write a random adjective-noun pair like this
   ```
   print("The",random.choice(adjectives)," ",random.choice(nouns))
   ```

   Spend a few minutes to create the three lists of words, and write a sentence that follows the pattern
   The <adjective> <noun> <verb> the <adjective> <noun>

Yay! With just a few lines of code, you can now generate …. how many?? …. sentences! (hint: a lot!)

Here is my solution:

```
import random
nouns = ["sun", "water", "fish", "cat", "grass"]
adjectives = ["pretty","ugly","red","smelly","fun"]
verbs = ["reads","eats","smells","drinks","grabs"]
print("The",random.choice(adjectives),random.choice(nouns),random.ch
oice(verbs),"the",random.choice(adjectives),random.choice(nouns))
```

And some fun output from this program
The smelly grass smells the fun water
The red cat grabs the fun sun

Assignment 3. Create a fun MadLib!

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. Open the CoolLists program that you created during class. Add a new piece of code (remember you go in the upper left corner, and click on +Code). Write a program that does the following:
   - Creates a list of persons you know - add at least five names
   - Creates a list of adjectives - add at least five adjectives
   - Creates a list of nouns (singular nouns) - add at least five nouns
   - Creates a list of verbs (this time, use the regular verb forms, like “eat”, “drink”) - add at least five verbs
   - Now write a MadLib story!
     You can choose your own, or here is a suggestion for a MadLib to use.
     Instead of each <word>, choose at random from the corresponding list. Have fun!

     Trip to the park
     Yesterday, <person> and I went to the park. One our way to the <adjective> park, we saw a <adjective> <noun> on a bike. We also saw big <adjective> balloons tied to a <noun>. Once we got to the <adjective> park, it started to <verb> and <verb>. We <verb> all the way home. Tomorrow we will try to go to the <adjective> park again, hope it does not <verb>.

Lesson 4. Loops. And how to do loooots of math.

What we will learn in this class:

- Go over the second assignment: (one possible) solution and improvements (as suggested by all of you!)
- Quick review: lists
- Learn about loops
- Putting it all together: Add all the numbers to 1000, and more!
- Assignment 4: Write a program to add a lot of numbers provided by the user

Review of the second assignment.

Remember the assignment guidelines

- The computer "thinks" of a random number between 1 and 100
  Hint:

```
import random
computerNumber = random.randint(1,7))
```

- The program asks the user to say a random number
- The program compares the two numbers, and if the computer number is larger, it says "I won", if the user number is larger, it says "You won"

One solution (there are many different correct solutions!)

```
import random
x = input ("Enter a number between 1 and 100: ")
computerNumber = random.randint(1,100)
print (computerNumber)
if (int (computerNumber) > int (x)) :
  print("I won!")
else:
  print("You won!")
```

Other great variations:

- Check if the user entered a number in the correct range

```
if (int (x) > int (100)) :
  print ("But you cheated!")
```

- Check if the user and computer tied

```
if(computerNumber > x):
 print("i won")
elif(computerNumber == x):
 print("we tied")
```

```
else:
 print("you won because you're amazing")
```

- Make the user win every time!

```
import random
computerNumber = random.randint(0,0)
x = input("Type a random number.")
if(int(x) > computerNumber):
 print("Ughhh! You won. You chose a bigger number then me :(")
else:
 print("Yesss! I won! My number was bigger then yours!")
print("My number was:", computerNumber)
```

Question: Why is the user winning every time?

Quick review. Lists.

Review questions:
How do we *declare* lists?
How do we write a list with 4 elements?
How do we access the second element in a list?

What will be the output of these programs?

```
myList = [3,10,4,12]
print(myList[0])
print(myList[2])
----
myList = [3,10,4,12]
myList.sort()
print(myList)
print(myList[0])
----
myList = [3,10,4,12]
myList.append(25)
myList.append([3,4])
print(myList)
----
myList = [3,10,4,12]
print(myList[0:2])
```

Trivia: Who was the first programmer ever?

Answer: Ada Lovelace! Read more about her here See below a picture of her, and the very first program, written by her.

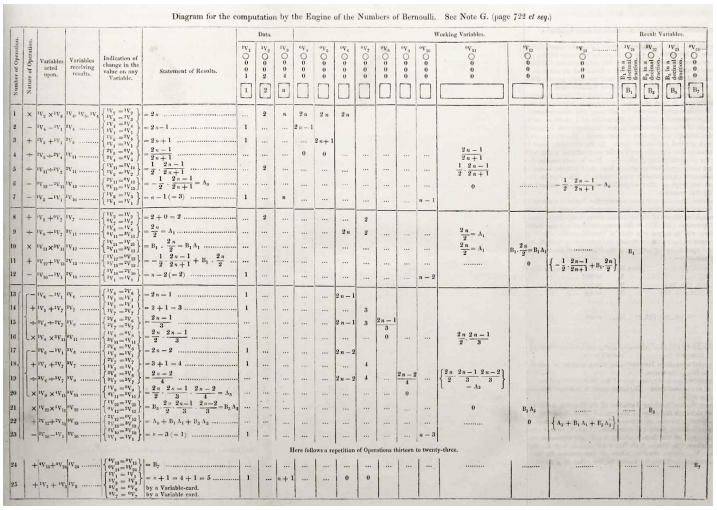


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

While Loops.

We've done a lot of cool things so far, but a lot of the things we have done were happening just once. And to be fair, while it's really cool that we now know how to write Python programs, we could have done all those math operations and if-then-else tests ourselves.

The big BIG power of computer programming comes from the fact that we can write programs that repeat a certain thing (math operation, word counting, anything you can think of) a million times, a billion times, a GAZILLION times, and do it quickly, much much faster than we as humans could do it.

Note: This to me is the major difference between humans and computers.

Humans are smart and slow.
Computers are non-smart and fast. (I always taught my kids that it's not nice to say "stupid" :)

How do loops work? Simply, these are statements that repeat what follows many times.

Let's look at *while* loops. (we'll look at *for* loops next time)

while(<condition>):
  do-something

The way we read this is: "as long as the <condition> is true, we do-something."
(you can also think of while-loops as an if-then-statement that is repeated many times)

Let's try to write a program that prints all the numbers from 1 to 10 (excluding 10)

```
number = 1
while(number < 10):
 print (number," ")
 number = number + 1
```

Now let's try to add all the numbers to 1000

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to LoopsLoopsLoops. Keep the ipynb extension -- remember this says this is a Python notebook.
3. First, we have to say our sum is 0 (we haven't added anything so far!), and that we start with number 1

   ```
   sum = 0
   number = 1
   ```
4. Now let's add a sum, to say that we want to add number to sum. Our program will look like this:

   ```
   sum = 0
   number = 1
   sum = sum + number
   print(sum)
   ```
5. Now let's make this repeat for all the numbers up to 1000 (and also print the result)

   ```
   sum = 0
   number = 1
   while(number < 10000):
    sum = sum + number
    number = number + 1
   print (sum)
   ```
6. How about summing up all the numbers up to 10000? How quickly you as a human could do it?

Let's try another while loop, now using a condition based on what the user is telling us. For instance, say we wanted to help the user with a lot of additions, not just one addition. As before, we want to ask for some numbers to add them up, but now we do it many times until the user tells us "stop"

7. Add a new piece of code (remember you go in the upper left corner, and click on +Code).

```
print("I will help you with a lot of math today")
signal = ""
while(signal != "stop"):
      a = input ("Tell me a number.")
      b = input ("Tell me another number:")
      print("This is the addition of those two numbers: ",
      int(a)+int(b))
      signal = input("Do you want more help? Say stop if you want to
      quit")
```

Assignment 4. Write a program that adds all the numbers that a user provides, until the user says "done"

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. Open the LoopsLoopsLoops program that you created during class. Add a new piece of code (remember you go in the upper left corner, and click on +Code). Write a program that does the following:
   - Tells the user you will help them add as many numbers as they want
   - Tells the user to say "done" when they are done providing numbers
   - Includes a while loop that checks if the user said "done"
   - Asks the user for numbers (inside the while loop), and adds them up
   - Stops when the user entered "done"
   - At the end, prints the sum of all the numbers entered by the user

Lesson 5. Loops (2). Many MadLibs. Rock, paper, scissors

What we will learn in this class:

- Go over the third assignment: (one possible) solution and improvements (as suggested by all of you!)
- Quick review: while loops

- Learn about more loops: *for*
- Putting it all together: Generate a looooot of sentences.
- Assignment 5: Write a Rock, Paper, Scissors game.

Review of the third assignment.
Remember the assignment guidelines

- Creates several lists: of persons you know, adjectives, nouns, verbs.
- Use the lists to write a MadLib story!
  You can choose your own, or use my suggestion for a MadLib to use.
  Instead of each <word>, choose at random from the corresponding list. Have fun!

  Here is a fragment:

```
import random

people = ["dad","mom","brother","sister","dog"]
adjectives = ["heavy", "light", "thin", "thick", "soft"]
noun = ["stick", "banana", "fire", "window", "tree"]
verb = ["smile", "burn", "float", "walk",  "laugh"]
#MadLib story

print("Yesterday", random.choice(people), "and I went to the
park. On our way to the", random.choice(adjectives), "park, we
saw a", random.choice(adjectives), random.choice(noun), "on a
bike.”)
```

  Output:

- *Yesterday dog and I went to the park. On our way to the thin park, we saw a thick tree on a bike. We also saw big soft balloons tied to a banana . Once we got to the light park, it started to laugh and laugh . We burn all the way home. Tomorrow we will try to go to the thick park again, hope it does not smile .*

Quick review. While Loops.

Review questions:
Why do we need loops?
How does a while loop work?
How many times does a while loop repeat? (or, when does a loop stop)

What will be the output of this program?

```
a = 1
while(a < 5):
 print ("Yay! ")
 a = a + 1

# this is what happens inside
# (note! this is a comment)
#a = 1
#"Yay! "        a = 2       a < 5?
#"Yay! "        a = 3       a < 5?
#"Yay! "        a = 4       a < 5?
#"Yay! "        a = 5       a < 5?
```

Why do we write?

```
a = a + 1
```

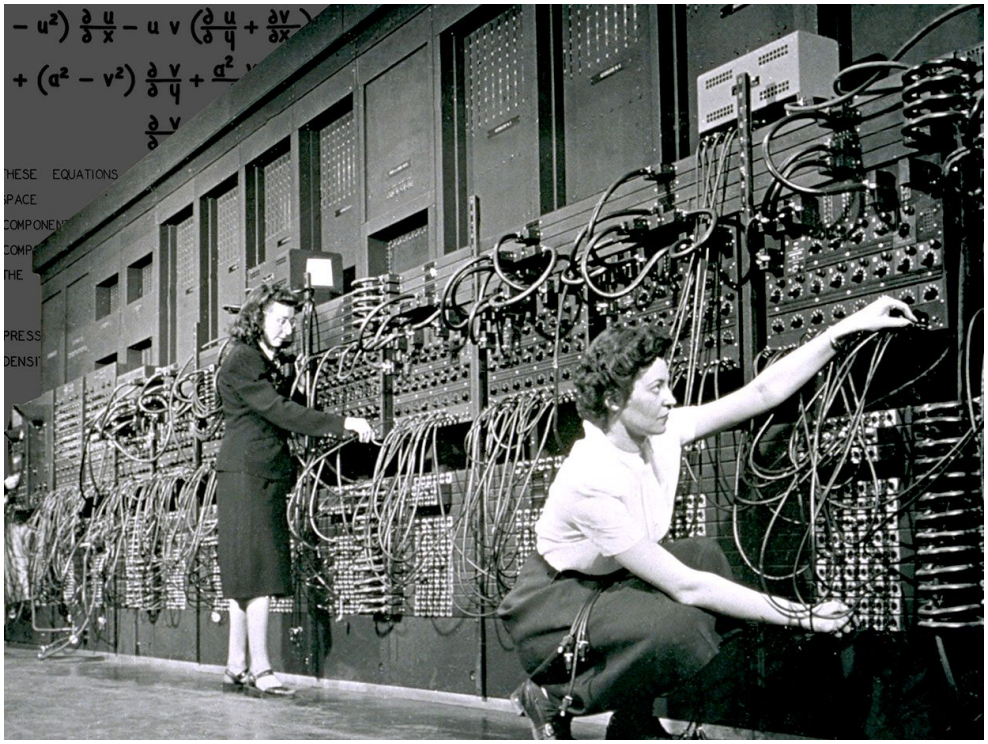
What will be the output of this?

```
a = 1
while(a < 5):
 print ("Yay! ")
```

How many times will this loop run?

```
stopSignal = "start"
while(stopSignal != "stop"):
 print ("I'm running")
 stopSignal = input("Do you want to run it again? Say stop if not.")
```

Trivia: What is in the picture below?



ENIAC (Electronic Numerical Integrator and Computer) was the first electronic general-purpose digital computer. Created in 1946. The people who operated the ENIAC were called "computers." Over time, the name ENIAC was lost, instead we use "computer" to refer to these computing machines.

For Loops.

We've learned about *while* loops so far, let's see another type of loop. For loops are very similar, except they loop through the elements of a list. They are as powerful as while loops, as they help us do a LOT of repetition. Why is repetition useful in programming?

Here is a for loop:

```
for x in list:
   do-something
```

The way we read this is: "do-something for every item x in the list"

Here's an example:

```
moods = ["happy", "sad", "cool", "relaxed", "angry"]
for a in moods:
 print ("I feel ", a)
```

If we want to repeat something a certain number of times, instead of writing down all the elements in a list, we can give a *range*

Let's try to write a program that prints 10 numbers

```
for i in range(10):
 print (i)
```

Big question: what will the program print?

Now let's try to add all the numbers to 10000, as we've done last time, but now let's do it with *for* loops:

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to MoreLoops. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Remember from last time. First, we have to say our sum is 0 (we haven't added anything so far!)
   ```
   sum = 0
   ```
4. Now let's make this repeat for all the numbers up to 10000 (and also print the result)
   ```
   sum = 0
   for number in range(10000):
    sum = sum + number
   print (sum)
   ```
5. Remember the while loop? Let's put our *while-loop* solution here, so we can look at them side by side
   ```
   sum = 0
   number = 1
   while(number < 10000):
    sum = sum + number
    number = number + 1
   print (sum)
   ```

Let's try another *for* loop, this time we can loop through all the words in a list. And do it twice!

6. Add a new piece of code (remember you go in the upper left corner, and click on +Code).
7. Let's do a small piece of the MadLib, but now we want many MadLibs!

```
import random
animals = ["cat","dog","spooky bird"]
moods = ["happy","sad","smelly"]

for i in range(10):
 print("The",random.choice(animals),"feels
very",random.choice(moods),"because it ate popcorn")
```

8. How about if we want to generate ALL the MadLibs we can for the lists above?
   For each animal, we want to go through all the moods.
   How do we do that? Nested loops!

I will get you started….

```
animals = ["cat","dog","spooky bird"]
moods = ["happy","sad","smelly"]

for myAnimal in animals:
  for a in moods:
      print("The",myAnimal,"feels very",a,"because it ate popcorn")
```

Assignment 5. Program a Rock, Paper, Scissors game. The game will run six times, and at the end you will write the score: how many times the computer won, how many times the user won.

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. Open the MoreLoops program that you created during class. Add a new piece of code (remember you go in the upper left corner, and click on +Code). Write a program that does the following:

- Tells the user a fun introductory message, and explains them the rules of Rock, Paper, Scissors
- Tells them they will play six times, and the player who wins more rounds is the winner.
- Write a *for* loop that repeat six times
- Inside the *for* loop:
    - The computer picks randomly one of [“R”, “P”, “S”]
    - Asks the user to make a choice, reminds them to use R for Rock, P for Paper, S for Scissors
    - Decide who wins (remember if-then-else)
    - Keep track of how many times the user won and the computer won
- Outside the for loop, at the end, reports the score, and congratulates the user if they won.
- Bonus! You can repeat the six-round game as many times as the user wants.

Lesson 6. Functions. Fun drawings with Turtle.

What we will learn in this class:

- Go over the fourth assignment: (one possible) solution and improvements
- Quick review: for loops
- Learn how to draw with Python
- Learn how to write functions in Python
- Putting it all together: Draw a pattern many times
- Assignment 6: Write a program to make a fun drawing!

Review of the fourth assignment.

Remember the assignment guidelines. Write a program that does the following:

- Tells the user you will help them add as many numbers as they want
- Tells the user to say “done” when they are done providing numbers
- Includes a while loop that checks if the user said “done”
- Asks the user for numbers (inside the while loop), and adds them up
- Stops when the user entered “done”
- At the end, prints the sum of all the numbers entered by the user

```
print("I will help you add as many numbers as you want.")
sum = 0
a = 0
while(a != "done"):
 sum = sum+int(a)
 a = input ("Tell me a number or say done if done saying numbers.")
print("The sum is:" ,sum)
```

Quick review. For Loops.

Review questions:
How does a for loop work?
How many times does a for loop repeat? (or, when does a loop stop)

What will be the output of this program?

```
for i in range(6)
 print ("Yay! ")
```

What will be the output of this program?

```
list = ["cat","dog","bird"]
for animal in list:
 print ("I have a", animal)
```

How about ...

```
list = ["cat","dog","bird"]
for animal in list:
 print ("I have a", animal)
 break
```

Note! You can stop a loop with the *break* command.
You can also “force” a loop to continue with the *continue* command

What will be the output of this program?

```
list = ["cat","dog","bird"]
for animal in list:
 continue
 print ("I have a", animal)
```

Trivia: The language of a computer consists of how many symbols?

Answer: Two symbols: 0 and 1

Bonus trivia question: how do we make numbers using these symbols? How about letters?

Humans	Computers
0	0
1	1
2	10
3	11

4 100
5 101
6 110
7 111
….

Same for letters, we use sequences of 0s and 1s
A 01000001

Bonus trivia question: If your fingers were computer bits, where you could either have your finger up (1) or down (0), up to what number can you count using 10 fingers?
Answer: 1023!

Drawing with Turtle.
Yeap! With a turtle! Not like a real turtle, but a library turtle.
Python has a library called Turtle that you can use to do all sorts of drawings. Let's try a few things out.

First off, you will need to make sure you have the library. The way we install a library is with the command *pip*

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to DrawingTurtle. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Write and run
   ```
   !pip3 install ColabTurtle
   from ColabTurtle.Turtle import *
   ```
4. Now the ColabTurtle library is available for you to use and do cool drawings

There are many things you can do. Here are some commands you can try.

Get a new pen (ie, initialize a new turtle). Set the speed of drawing to 10 (you can set it to anything you want)
```
initializeTurtle()
```

Set the color of the pen
```
color('orange')
```

Set the color of the background

```
bgcolor('white')
```

Note: valid colors are: `'white'`, `'yellow'`, `'orange'`, `'red'`, `'green'`, `'blue'`, `'purple'`, `'grey'`, `'black'`

Set the width of the line

```
width(3)
```

Go forward or backward by a certain number of units (200 is approx 1 inch)

```
forward(200)
backward(100)
```

Go left or right by a certain number of degrees

```
left(45)
right(135)
```

Move the turtle to certain coordinates

```
setx(100)
sety(80)
goto(100,80)
```

Show or hide the turtle

```
showturtle()
hideturtle()
```

Set the speed of the turtle

Put the pen down or up

```
pendown()
penup()
```

Let's try doing some fun drawings -- we can draw a star!

5. Go into your DrawingTurtle program. Add one new piece of code by clicking on +Code in the upper left corner (make sure you run the pip3 and import commands from above)
6. Try this code

```
initializeTurtle()
speed(10)
color('red')
```

```
bgcolor('white')
width(1)
for i in range(36):
    forward(250)
    left(170)
```

Functions.
Oftentimes, we write code that we then need again at a later time. In programming, a nice way to organize our code is through the use of functions, which allow us to write something once, and then reuse many times.

We define a function simply by using the keyword *def*

```
def function(input1, input2, ...):
  our code here that implements what the function does
  [optiona] return output
```

To see what functions can do for us, and also to see an example, assume we want to draw the nice red star we drew earlier three times, at different coordinates.

One way we could do this is as below:

```
initializeTurtle()
speed(10)
color('red')
bgcolor('white')
width(1)
#one star
penup()
goto(300, 300)
pendown()
for i in range(36):
    forward(250)
    left(170)
#a second star
penup()
goto(500, 400)
pendown()
for i in range(36):
```

```
    forward(250)
    left(170)
#a third star
penup()
goto(200, 600)
pendown()
for i in range(36):
    forward(250)
    left(170)
```

It works, and we have three nice stars. But this is not very efficient, because I just keep writing the same code many times. What if I want to draw 100 stars? Or, what if I decide I want to change the number of units I go forward from 250 to 150? I need to make that correction many times.

Another way to do it is by using functions. We define a function -- let's call it *star* -- and then we just use that function whenever we want a star

```
#Notice how I use x and y as variables - and I can set them to
#anything I want when I call the function!

def star(x, y):
 penup()
 goto(x, y)
 pendown()
 for i in range(36):
     forward(250)
     left(170)

# Now let me draw three stars, at different (x,y) coordinates
initializeTurtle()
speed(10)
color('red')
bgcolor('white')
width(1)

star(300,300)
star(500,400)
star(200,600)
```

Let's try to write a function that does a square:

7. Add a new piece of code (use +Code in the upper left corner)
8. Define a function called square, which receives as input the coordinate, and the length of one side

```
bgcolor('white')def square(x, y, side):
 penup()
 goto(x, y)
 pendown()
 for i in range(0,4):
   forward(side)
   left(90)
```

9. Try drawing a square

```
# initialization steps
initializeTurtle()
speed(10)
color('red')

width(1)

# draw two squares
square(200,200,50)
square(100,100,75)
```

Assignment 6. Make a cool drawing to impress your parents and your teacher. The only requirements are that you use at least one loop, and at least one function. For the rest, the sky's the limit :)

Lesson 7. Reading from web pages. How many words in this book?

What we will learn in this class:

- Go over the fifth assignment: (one possible) solution and improvements
- Quick review: functions
- Learn how to read from webpages (and from regular files)

- Putting it all together: How many words in this book?
- Assignment 7: Write a first version of your project.

Review of the fifth assignment.
Remember the fifth assignment was a Rock / Paper / Scissors game. Write a program that does the following:

- Tells the user a fun introductory message, and explains them the rules of Rock, Paper, Scissors
- Tells them they will play six times, and the player who wins more rounds is the winner.
- Write a *for* loop that repeat six times
- Inside the *for* loop:
  - The computer picks randomly one of [“R”, “P”, “S”]
  - Asks the user to make a choice, reminds them to use R for Rock, P for Paper, S for Scissors
  - Decide who wins (remember if-then-else)
  - Keep track of how many times the user won and the computer won
- Outside the for loop, at the end, reports the score, and congratulates the user if they won.

Here is one possible solution:

```
import random
moves = ["R","P","S"]
print("Hello, we are now playing a few rounds of rock paper scissors. The
rules are simple, rock beats scissors, scissors beats paper, and paper
beats rock")
print("The player who won more rounds wins")
Player_score = 0
Computer_score = 0

for i in range(6):
 Player = input("chose R , S or P: ")

 Computer = random.choice(moves)
 if(Player == Computer):
   print("It's a tie")
 if(Player == "S"):
  if(Computer == "R"):
   Computer_score = Computer_score + 1
```

```
    print("Computer won")
   elif(Computer == "P"):
    Player_score = Player_score + 1
    print("You won")

 if(Player == "P"):
  if(Computer == "S"):
   Computer_score = Computer_score + 1
   print("Computer won")
  elif(Computer == "R"):
   Player_score = Player_score + 1
   print("You won")

 if(Player == "R"):
  if(Computer == "P"):
   Computer_score = Computer_score + 1
   print("Computer won")
  elif(Computer == "S"):
   Player_score = Player_score + 1
   print("You won")

if(Player_score > Computer_score):
print("You won",Player_score,"rounds, and the computer
won",Computer_score,".So you won more rounds then the computer.")
elif(Player_score < Computer_score):
 print("You won",Player_score,"rounds, and the computer
won",Computer_score,".So the computer won more rounds then you.")
elif(Player_score == Computer_score):
print("We have a tie")
```

Quick review. Functions.

Review questions:
Why do we use functions?
What does a function do?

What will be the output of this program?

```
def addNumbers(a,b):
 return (a+b)

print(addNumbers(3,7))
```

What will this program do?

```
def isHappy(word):
 if word in ["happy","fine","amused"]:
   return 1
 else:
   return 0

def isSad(word):
 if word in ["sad","terrible","annoyed"]:
   return 1
 else:
   return 0

userMood = input("How are you today?\n")
if(isHappy(userMood)):
 print ("Glad to hear that")
elif(isSad(userMood)):
 print ("I'm sorry to hear that")
```

What will this program do?

```
def drawLine(x,y):
 penup()
 goto(x,y)
 pendown()
 color(random.choice(["red","blue","green"]))
 forward(100)
```

```
initializeTurtle(10)
for i in range(30):
 x = random.randint(0,800) # picks random no. between 0-800
 y = random.randint(0,500)
 drawLine(x,y)
```

PROJECTS!!
What project will you work on?

Project ideas:

- interact with user to get input on what drawing to do. Make a drawing based on user input (eg, forest, beach, different colors, etc.)
- “guess the animal” game by asking user to think of an animal, then the computer will ask questions (“does it have stripes?”, “is it big?”, etc.) Maybe try for all animals.
- “guess the number” game where the user thinks of a number, then the computer tries to guess it by asking questions / making statements
- hangman, where the computer thinks of a word, then the user tries to guess it.
- animation of the rocket launch
- a function to make your own flag
- riddle game with several riddles that the computer will ask the user

Reading/writing from a file. Reading from a webpage.
So far we only used information provided by a user, or information that we included in our programs. Often time, we need to process information that is stored in documents (or files). How can we have access to that?

It’s quite easy! We simply indicate where we want to get our information from, and then we store that information in some variables so we can process it inside our program.

I will tell you how to read/write from files on your computers, but we are not going to use that with the Colab environment (remember, with Colab, our code is “in the cloud”, on the Google

computers, and accessing the files on our computers requires a few extra steps we will not cover here). Nonetheless, here are the main steps to read/write from a file.

Read:

```
file = open("myfile.txt","r")
lines = file.readlines()
```

Now in `lines` we have a list with all the lines in my file! So I can print them out:

```
for line in lines:
   print (line)
```

Write:

```
file = open("myfile.txt","w")
file.write("This is a line")
file.write("This is another line")
file.close()
```

How about reading from a file that is already online -- like a webpage?
For instance, here is one file online

http://www.gutenberg.org/files/236/236-0.txt

We can read from an URL with the help of a library called BeautifulSoup

```
import bs4
import requests
URL = "http://www.gutenberg.org/files/236/236-0.txt"
book = requests.get(URL, {}).text
```

Now in the variable `book` we have the entire content of a book, from that URL! You can try reading other pages that you like, and using them in your programs to do all sorts of interesting things.

Let's see a few things we can do with a text (string)...

We can split a string into words:

```
words = book.split(" ")
```

We can then count all the words:

```
print (len(words))
```

Yes, you just counted all the words in an entire book!

We can count how many times one word appears:

```
print (words.count("Mowgli"))
```

What is a word that it's even more frequent than "Mowgli"?

Try counting some words from a webpage yourself!

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to ManyWords. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Try

```
import bs4
import requests
URL = "http://www.gutenberg.org/files/236/236-0.txt"
# this is the Jungle Book on Project Gutenberg
book = requests.get(URL, {}).text
words = book.split(" ")

print (len(words))
print (words.count("Mowgli"))
```

4. Try also this alternative of splitting a book into words, which separates words based on space as well as punctuation

```
import bs4
import requests
import re
URL = "http://www.gutenberg.org/files/236/236-0.txt"
book = requests.get(URL, {}).text

words = re.split('[ ,-?!:()]',book)  #split based on space or punct.

print(len(words))
words.count("Mowgli")
```

Assignment 7.

- *Write a first version of your project. Save it in your <YourName>FunCode* folder.
- *[Optional] Try a few more things when reading a webpage. For instance, you can try counting all the words in the file. Or counting all the words that differ from ["the","of","a","an","with"] . Or finding the most frequent words on a webpage.*

Lesson 8. Dictionaries. Creating a secret code.

What we will learn in this class:

- Go over the sixth assignment: super cool drawings with Turtle
- Quick review: files and webpages
- Learn how to create a dictionary structure
- Putting it all together: Translate a text into a secret message
- Assignment 8: Final project!

Review of the sixth assignment.
Make a cool drawing to impress your parents and your teacher. The only requirements are that you use at least one loop, and at least one function. For the rest, the sky's the limit :)

Here is a function used in one of the solutions.

```
def tree(branchLen):
    if branchLen > 2:
        forward(branchLen)
        right(20)
        tree(branchLen-20)
        left(40)
        tree(branchLen-20)
        right(20)
        backward(branchLen)
```

Note how the function `tree` calls the function `tree`? This is called *recursivity* - it is an important concept in programming, as it allows you to solve very complicated problems in simple ways. For instance here, we can think of a tree as being composed of multiple trees, which in turn are composed of multiple trees, and so on. So even if the tree looks complicated, with lots and lots of branches, it is actually just a few lines of code.

Quick review. Functions.

Review questions:
Why do we need to use files (documents) to read and write from our programs?
How do we read from a file?
Can we read from a webpage?

What happens when I run this code?

```
URL = "http://mysite.org/mypage.txt"
myContent = requests.get(URL, {}).text
```

How about when I run this code?

```
words = myContent.split(" ")
```

Or this one?

```
print (len(words)) #len means length
print (words.count("today"))

myList = ["Ben", "water", "idk", "here", "Ben", "idk"]
```

Dictionaries.

We have seen different ways of storing data using Python. Can you name the ways in which we have stored data so far?

There is another very convenient way of storing data which is a dictionary. As the name says, you can think of it as a dictionary, in which each word is a "key" to a definition.

Here is one dictionary:

```
mydict = {"water":"a kind of liquid", "dog":"an animal with four legs",
"food":"something you can eat"}
mylist = ["water","dog","food"]
print(mylist[1])
```

You see it has some similarities with the list, except that each item has two parts: the *key* (in this example the word), and the *value* (in this example, the definition).

Can you spot another difference with respect to lists? (hint: remember what type brackets we used when declaring a list?)

Here are some other examples of a dictionary:

```
person = {"name":"Rada", "likes":"programming and reading", "address":"Ann Arbor"}
```

Here, we want to have entries for the attributes of a person, and the corresponding values.

Or another one:

```
secretCode = {"a":"c","b":"d","c":"e"}
```

Here, we want to create a secret code, so we write a “map” between original characters and secret encodings (see below for more)

What can we do with a dictionary? Lots of things.

Very much like a list, we can access elements in the dictionary. Recall that in a list we used an “index” (position in the list) to access an element. Here, we can simply use the key of an item to access it. For instance,

```
mydict = {"water":"a kind of liquid", "dog":"an animal with four legs", "food":"something you can eat"}

print (mydict["dog"])
```

will print the definition I included for “dog” in the dictionary I defined earlier.

We can also loop through all the elements in a list:

```
for word in mydict:
 print (word," ",mydict[word])
```

will display all the entries in the mydict dictionary, the word followed by its definition

We can also loop through all the elements in a list using both the key and the value:

```
for attribute, value in person.items():
 print(attribute," ",value)
```

will display all the entries in the person dictionaries.

We can add an entry from the dictionary, simply by creating a new key:

```
secretCode = {"a":"c","b":"d","c":"e"}
```

```
secretCode["d"] = "f"
```

We can remove an entry from a dictionary:

```
del secretCode["d"]
```

We can count the number of entries in a dictionary:

```
print(len(secretCode))
```

We can get all the keys or all the values

```
person.keys()
person.values()
```

What will this code do?

```
for word in sorted(dictionary.keys()):
 print (word," ",dictionary[word])
```

Let's try to write a program that does a Caesar cipher! This is a way of creating secret messages, so that only you and the person who knows "the rule" (how you created the secret message) can decipher. It's a very simple idea, we basically replace each character in the alphabet with another character some fixed number of positions down the alphabet.

1. Go to drive.google.com
   Go into your <*YourName*>*FunCode* folder. It is important that you keep everything in the same folder, so I can help you as needed.
2. On the left side of the browser, go to New / More / Google Colaboratory. In the upper left corner, click on Untitled0 and change to Dictionary. Keep the ipynb extension -- remember this says this is a Python notebook.
3. Let's use a dictionary to create a Caesar cipher

```
caesarCipher = {'A': 'D', 'C': 'F', 'B': 'E', 'E': 'H', 'D': 'G',
'G': 'J', 'F': 'I', 'I': 'L', 'H': 'K', 'K': 'N', 'J': 'M', 'M':
'P', 'L': 'O', 'O': 'R', 'N': 'Q', 'Q': 'T', 'P': 'S', 'S': 'V',
'R': 'U', 'U': 'X', 'T': 'W', 'W': 'Z', 'V': 'Y', 'Y': 'B', 'X':
'A', 'Z': 'C', 'a': 'd', 'c': 'f', 'b': 'e', 'e': 'h', 'd': 'g',
'g': 'j', 'f': 'i', 'i': 'l', 'h': 'k', 'k': 'n', 'j': 'm', 'm':
'p', 'l': 'o', 'o': 'r', 'n': 'q', 'q': 't', 'p': 's', 's': 'v',
'r': 'u', 'u': 'x', 't': 'w', 'w': 'z', 'v': 'y', 'y': 'b', 'x':
'a', 'z': 'c'}
```

What is the rule used in this cipher?

4. Let's create a function that takes a string (a text) and a dictionary that includes a Caesar cipher, and creates a secret message.

```
def encode(message,secretEncoding):
 secretMessage = "" #the secret message is at first empty

 for c in message:  #go through message one character at a time
   if c in secretEncoding.keys():   # make sure the character is in
the dictionary
     secretMessage = secretMessage + secretEncoding[c]
   else:
     secretMessage = secretMessage + c
 return secretMessage
```

5. Now let's use our function to create secret messages!

```
text = "My name is Rada"
mySecretText = encode(text,caesarCipher)
print (mySecretText)
```

Assignment 8.

- *Finish your project! Please let me know if you would like to meet for 30 min during the coming week so I can help you with your project.*
- *[Optional] Write a decoder for secret messages. Very much like the encoder, except that it works in the reverse: it takes a secret message, and it tells you the original message. Hint: you can "reverse" a Caesar cipher with one line, for instance:*

```
reversedCaesarCipher = dict((v, k) for k, v in caesarCipher.items())
```

What's next? Ways to continue learning.

You now know the basics of computer programming in Python! You've learned some of the most important concepts in programming: variables, lists, and dictionaries, conditionals, loops, functions, and input/output. You have already seen how you can use what you learned for a lot of cool projects, and there are sooo many other projects you can do with this newly acquired knowledge. For instance, you could create a conversation system, or you could write a program

that generates Haiku poetry or cooking recipes, or you could develop a dice game or a card game, or you could write a system that analyzes the online reviews of a product and determines if people like or dislike the product. There are so many amazing things you can accomplish now that you know that you can add all the numbers to a billion or count all the words in a book in a matter of seconds.

There are of course many more things to learn. For instance, we haven't addressed at all concepts such as recursivity (when a function calls itself), or object oriented programming (when we define objects that have certain properties and behaviors). We also talked only a little about strategies to solve problems (algorithms).

There are many ways in which you can continue to learn. Here are a few you can try:

- Codecademy
- Datacamp for Python
- Several courses offered on Coursera

But the very best way to learn is to program! Pick a problem that you are motivated to solve, and just get started. When you get stuck, ask around, look online, and you will find solutions. And on the way to solving the problem, you will learn a lot! I am so excited to see what you will accomplish next as brand-news computer programmers!