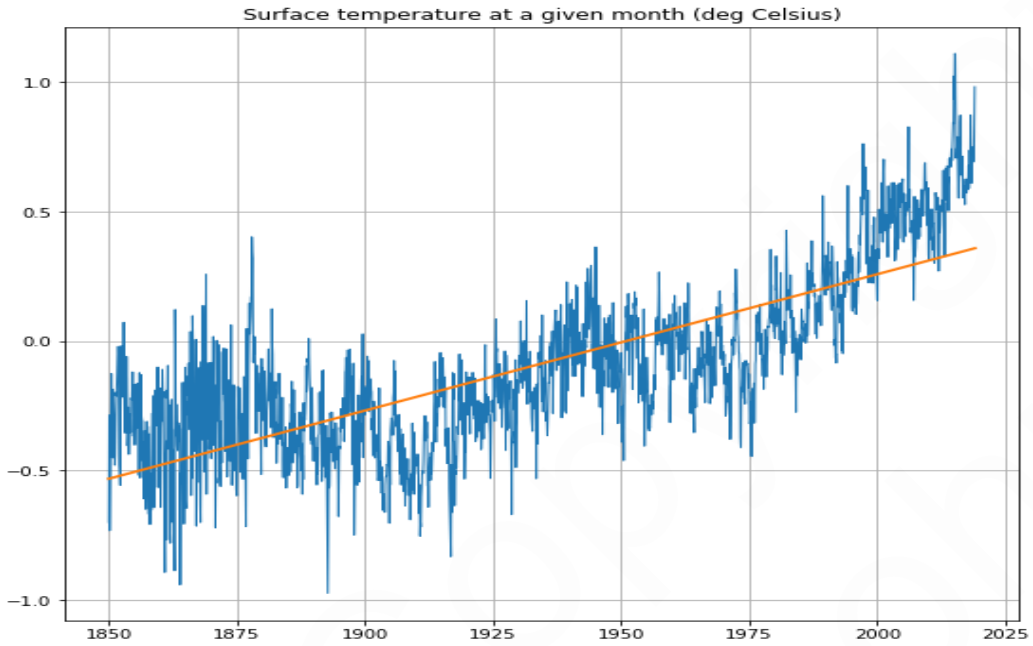
Climatic Applications Using Python




```
# Part II: trend of a gmsta
timeVar=tGMSTA[0:numMonths-11]
values=temperature1D[0:numMonths-11]
time,trendGMSTA=stat.LinearTrend(timeVar, values)

### Example I:  Displaying gmsta.
#---------------------------------------------------------------
# cerat figure, axes instances
fig1=plt.figure()
plt.plot(tGMSTA[0:numMonths-11], temperature1D[0:numMonths-11], \
         linewidth = 1 )
plt.plot(time,trendGMSTA,linewidth=2)
plt.grid()
plt.title('Surface temperature at a given month (deg Celsius)')
plt.gcf().set_size_inches(9,9)
plt.show()
##================================================================
```

Prof. Zhaohua Wu

Department of Earth, Ocean, and Atmospheric Science

Florida State University

OUTLINE

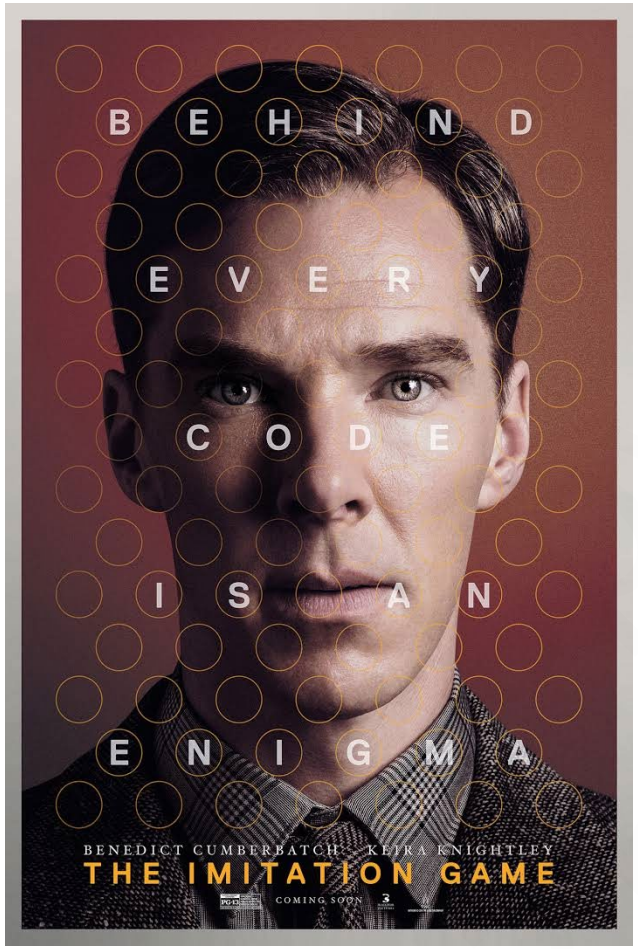
- **Computer and Modern Computing**
- **Preliminary Python Resources**
- **Basic Python Scripting Language Structure**
- **An Example**

FATHER OF THE MODERN COMPUTER SCIENCES

Alan Turing, 23 June 1919 – 7 June 1954, was a mathematician, logician, cryptanalyst, and computer scientist.

Turing's homosexuality resulted in a criminal prosecution in 1952—homosexual acts were illegal in the United Kingdom at that time—and he accepted treatment with female hormones, and chemical castration, as an alternative to prison. He died in 1954, several weeks before his 42nd birthday, from an apparently self-administered cyanide poisoning

THE IMITATION GAME



Alan Turing - Celebrating the life of a genius

Cambridge University
4 years ago • 141,475 views
Saturday 23 June 2012 marks the centenary of the birth of **Alan Turing** - mathematical genius, hero of the WWII code breakers of ...

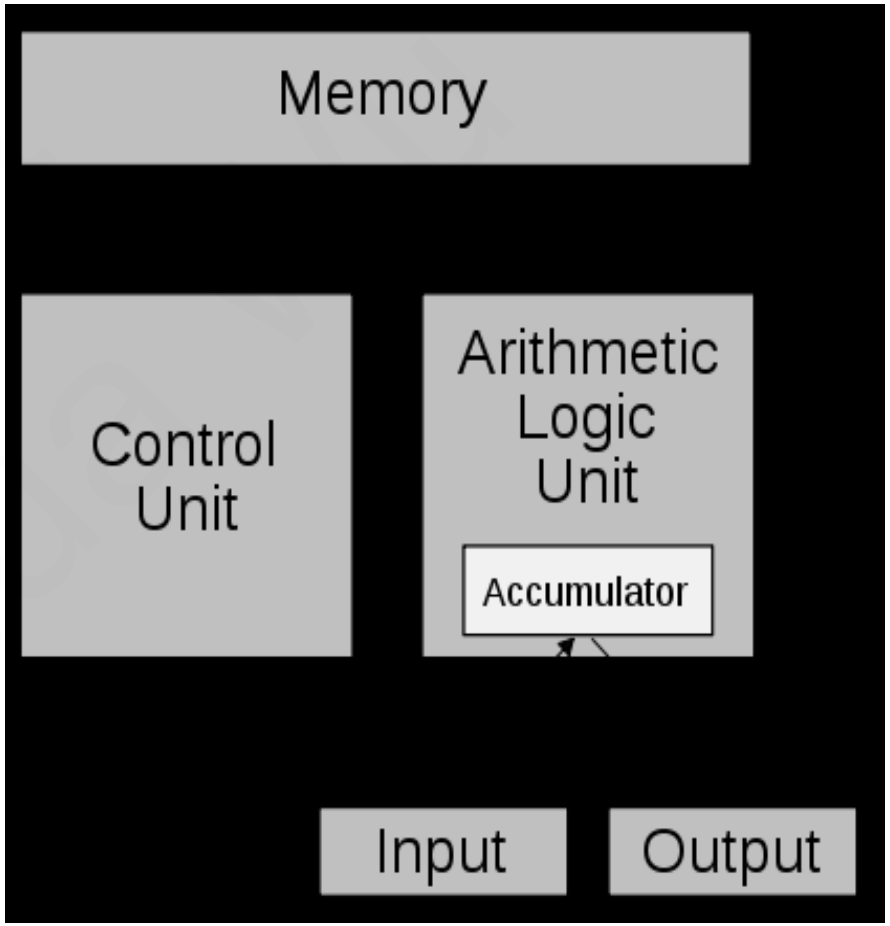
Breaking the Code: Biography of Alan Turing (Derek Jacobi, BBC, 1996)

Ciencias Cognoscitivas
5 years ago • 681,478 views
A biography of the English mathematician **Alan Turing**, who was one of the inventors of the digital computer and one of the key ...
CC

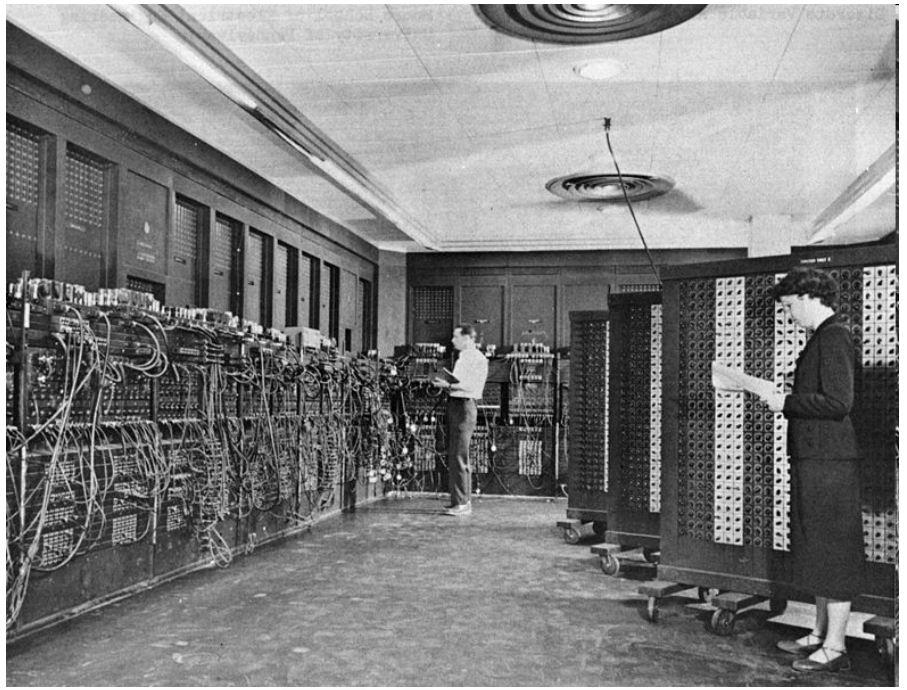
ALAN TURING

Joe Stoltz
5 years ago • 169,363 views
A brief video biography of the achievements of **Alan Turing**.

JOHN VON NEUMANN

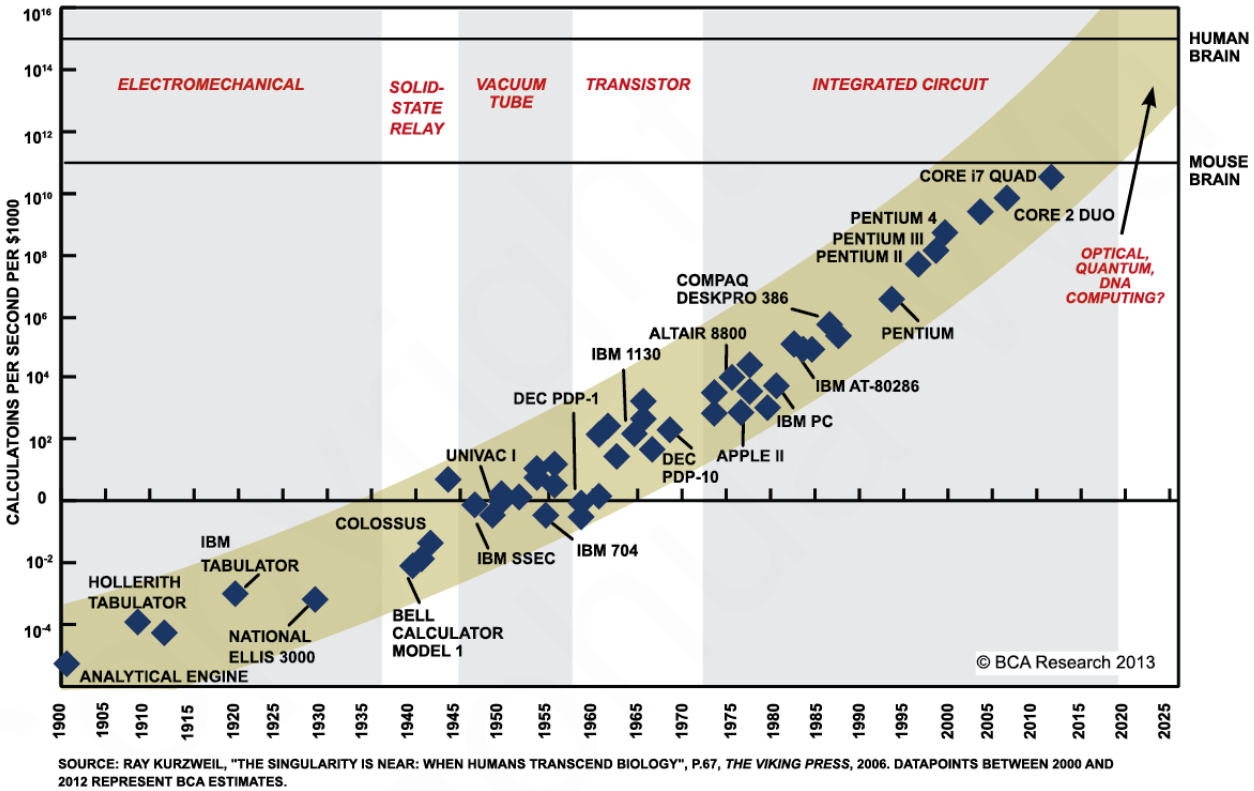


ENIAC



ENIAC, short for **Electronic Numerical Integrator And Computer**, was the first general-purpose electronic computer. It was a Turing-complete, digital computer capable of being reprogrammed to solve a full range of computing problems. ENIAC was designed to calculate artillery firing tables for the U.S. Army's Ballistic Research Laboratory, but its first use was in calculations for the hydrogen bomb.

MOORE'S LAW



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

- **Moore's law** describes a long-term trend in the history of computing hardware, in which the number of **transistors** that can be placed inexpensively on an integrated circuit has doubled approximately every two years.

QUANTUM COMPUTER

- In 1982, the Nobel prize-winning physicist **Richard Feynman** thought up the idea of a 'quantum computer', a computer that uses the effects of quantum mechanics to its advantage.
- One such development was the invention of an algorithm to factor large numbers on a quantum computer, by **Peter Shor** (Bell Laboratories). By using this algorithm, a quantum computer would be able to crack codes much more quickly than any ordinary (or classical) computer could. In fact a quantum computer capable of performing Shor's algorithm would be able to break current cryptography techniques in a matter of seconds.
- With the motivation provided by this algorithm, the topic of quantum computing has gathered momentum and researchers around the world are racing to be the first to create a practical quantum computer.

MOORE'S LAW VS. NEVEN'S LAW

- Following Moore's law, with N numbers of transistors, it can produce the following number of different arrangements of 1's and 0's

$$2^N$$

- In quantum computing, based on Neven's law, if a computer has N qbits, it can produce the following number of different arrangements of 1's and 0's

$$2^{2^N}$$

N	2	5	10
Moore	4	32	1024
Neven	16	4,294,967,295	$10^{308.25471556}$ (309 digits)

QUANTUM COMPUTER

UPFRONT 15 November 2017

IBM has built two new quantum computers with 20 and 50 qubits

IBM has announced a leap forward in quantum computing. The company unveiled two new quantum computers, one with 20 quantum bits, or qubits, which IBM says will be available at the end of the year, and a 50-qubit prototype.

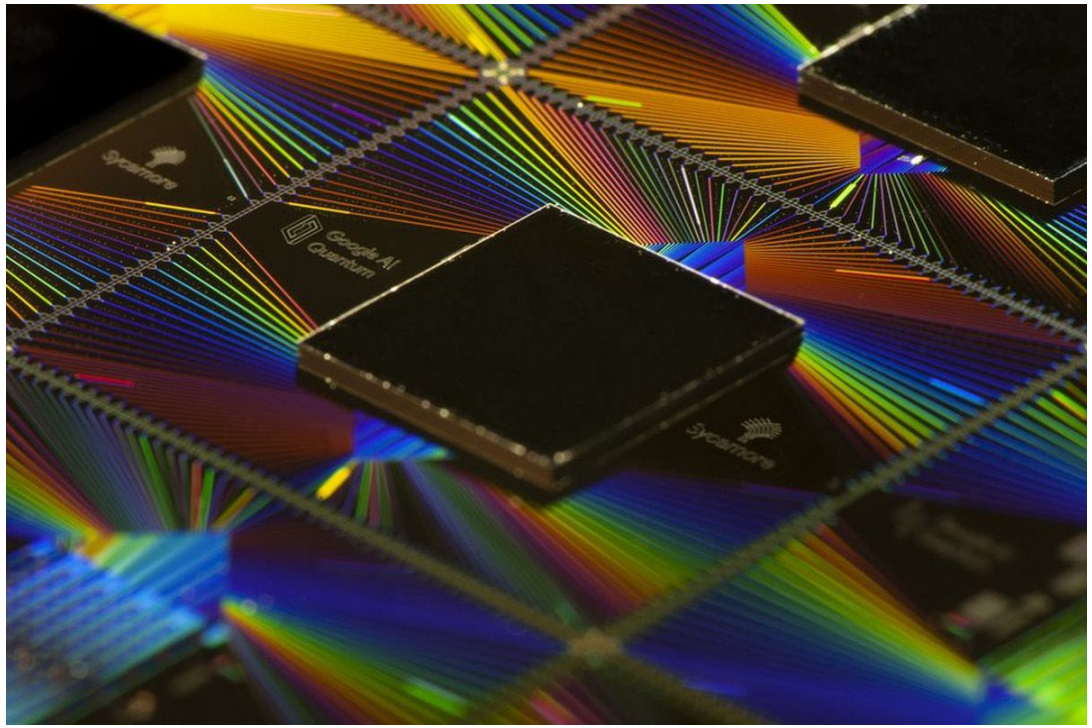
IBM is racing Google and several other firms to establish quantum supremacy by outperforming classical computers with quantum machines. Right now, classical computers can simulate 56 qubits.

Both of these new quantum computers can perform quantum calculations for 90 microseconds, longer than any other quantum computer to date. The 20-qubit computer will be available to IBM's clients at the end of the year for running quantum algorithms and simulations. The 50-qubit prototype is not yet ready for widespread use, but IBM says it will form the basis of their future quantum computers.

> **"Both computers can do quantum calculations for 90 microseconds, longer than any other to date"**

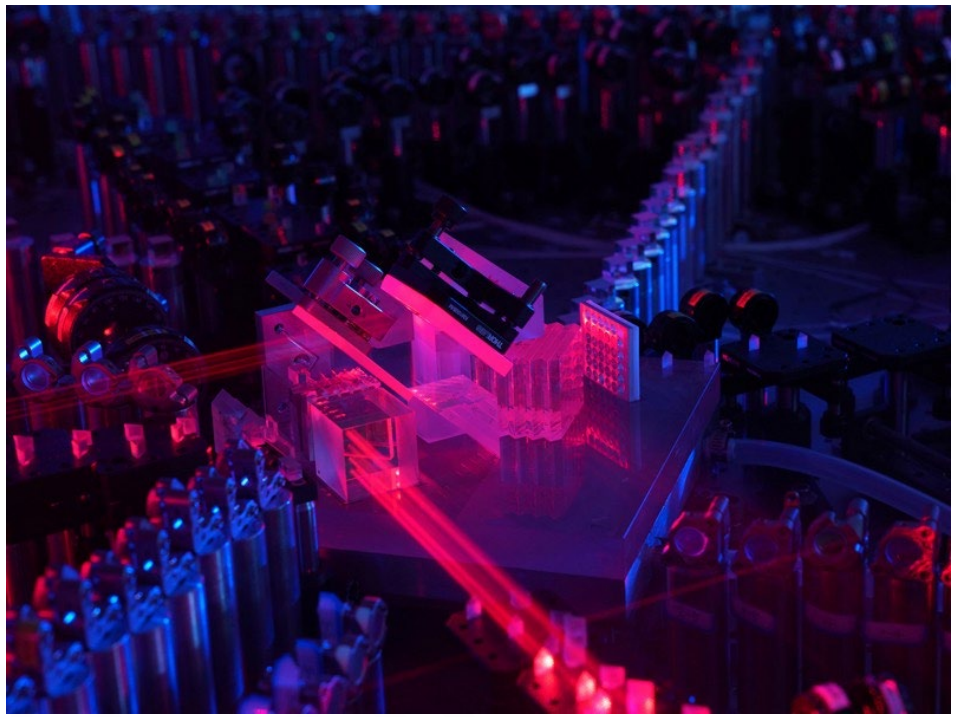
This article appeared in print under the headline "Qubits for hire"

2019 QUANTUM SUPREMACY OF GOOGLE



- Google says that its 54-qubit Sycamore processor was able to perform a calculation in 200 seconds that would have taken the world's most powerful supercomputer 10,000 years.

2020 QUANTUM SUPREMACY OF CHINA



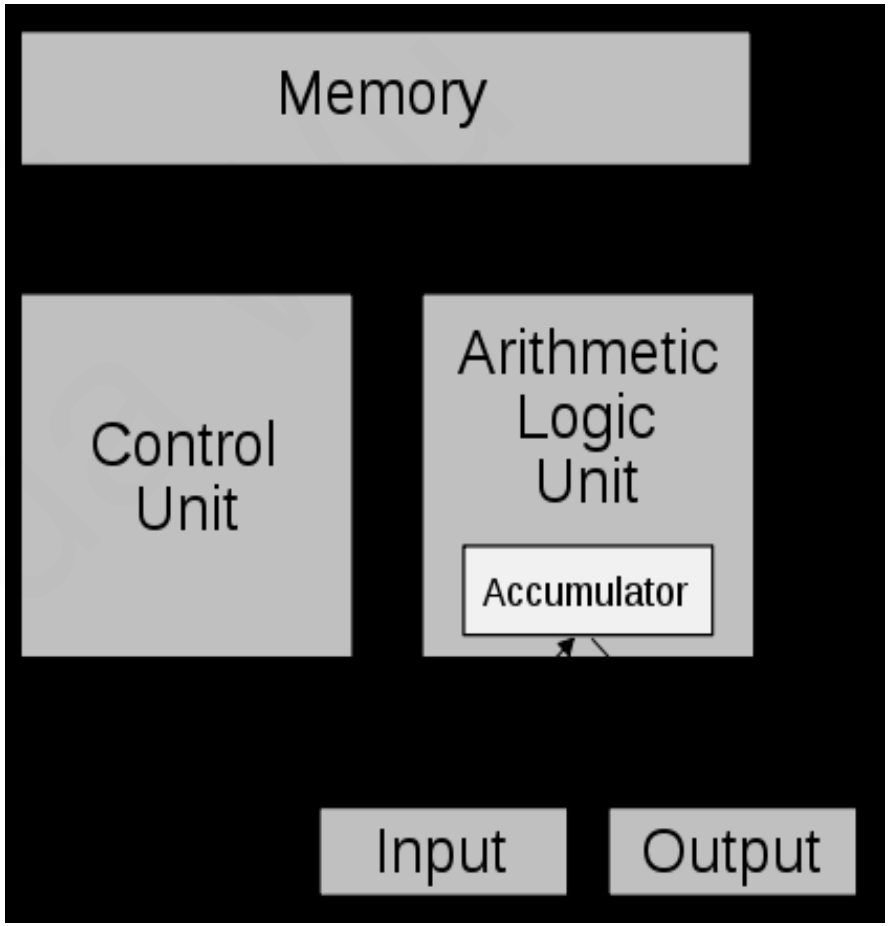
- A research team of the University of Science and Technology of China found solutions to the boson-sampling problem in 200 seconds. They estimate these would take 2.5 billion years to calculate the world's most powerful supercomputer, about 10^5 times faster than the Google one.

ARTIFICIAL INTELEGENCE

- IBM's deepblue (1997)

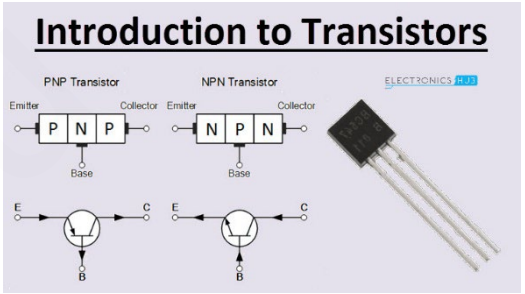
- Google Alphago (2017)

JOHN VON NEUMANN



FROM 0's & 1's TO A FULL CONTROL OF ELECTRONIC FLOW

- A transistor is a miniature semiconductor that regulates or controls current or voltage flow in addition amplifying and generating these electrical signals and acting as **a switch/gate** for them, i.e., producing ones and zeros.



- A combination of ones and zeros and a way to interpret the combination makes it possible to control output from a given input. The latter forms the basics of a computer language.

0	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0

PROGRAMMING LANGUAGES

- Programming languages provide various ways of specifying programs for computers to run.
- Unlike **natural languages**, programming languages are designed to permit no ambiguity and to be concise. They are purely written languages and are often difficult to read aloud.
- They are generally either translated into **machine code** by a **compiler** or an **assembler** before being run, or translated directly at run time by an **interpreter**.
- There are thousands of different programming languages—some intended to be general purpose, others useful only for highly specialized applications.

SOME PROGRAMMING LANGUAGES

Lists of programming languages	Timeline of programming languages, List of programming languages by category, Generational list of programming languages, List of programming languages, Non-English-based programming languages
Commonly used Assembly languages	ARM, MIPS, x86
Commonly used high-level programming languages	Ada, BASIC, C, C++, C#, COBOL, Fortran, Java, Lisp, Pascal, Object Pascal
Commonly used Scripting languages	Bourne script, JavaScript, **Python**, Ruby, PHP, Perl

OUTLINE

- Computer and Modern Computing
- **Preliminary Python Resources**
- Basic Python Scripting Language Structure
- An Example

TEXTBOOKS ON PYTHON

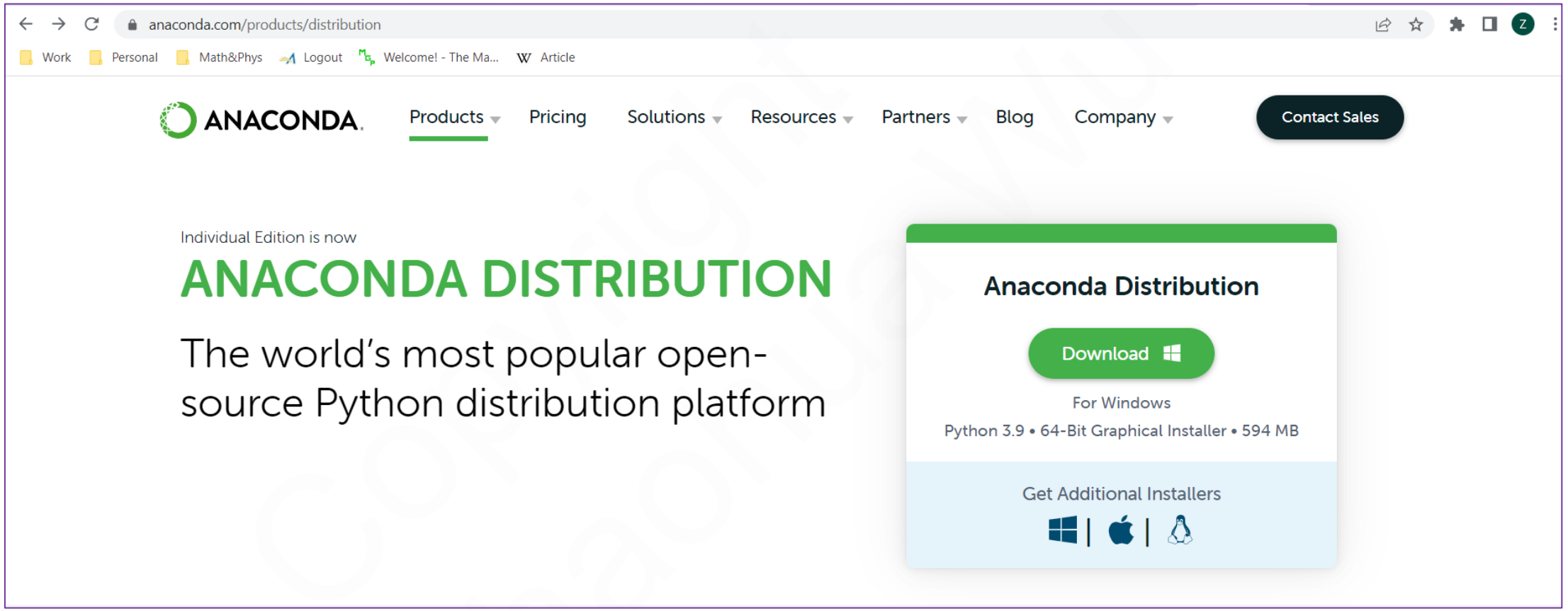
(1) A Byte of Python, by Swaroop C. H. (https://python.swaroopch.com/)

(2) Introduction to Scientific Computing in Python, by Robert Johansson, 2016 (https://raw.githubusercontent.com/jrjohansson/scientific-python-lectures/master/Scientific-Computing-with-Python.pdf)

(3) How to Think Like a Computer Scientist: Learning with Python. By Allen Downey, Jeffrey Elkner, and Chris Meyers, 2008, Green Tea Press. (https://runestone.academy/ns/books/published/thinkcspy/index.html)

(4) Python Programming and Visualization for Scientists, by Alex J. DeCaria, 2016, Sundog Publishing

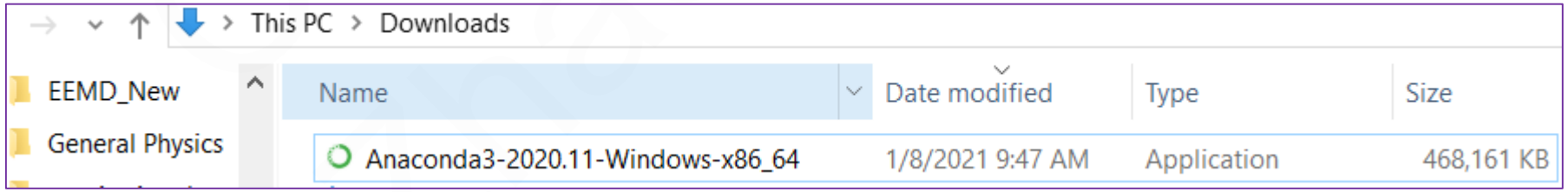
DOWNLOAD


anaconda.com/products/distribution
Work
Personal
Math&Phys
Logout
Welcome! - The Ma...
Article
ANACONDA.
Products
Pricing
Solutions
Resources
Partners
Blog
Company
Contact Sales
Individual Edition is now
ANACONDA DISTRIBUTION
The world's most popular open-source Python distribution platform
Anaconda Distribution
Download
For Windows
Python 3.9 • 64-Bit Graphical Installer • 594 MB
Get Additional Installers

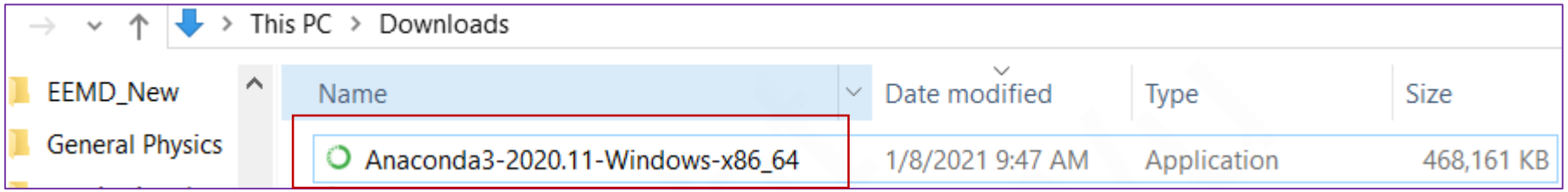
DOWNLOAD

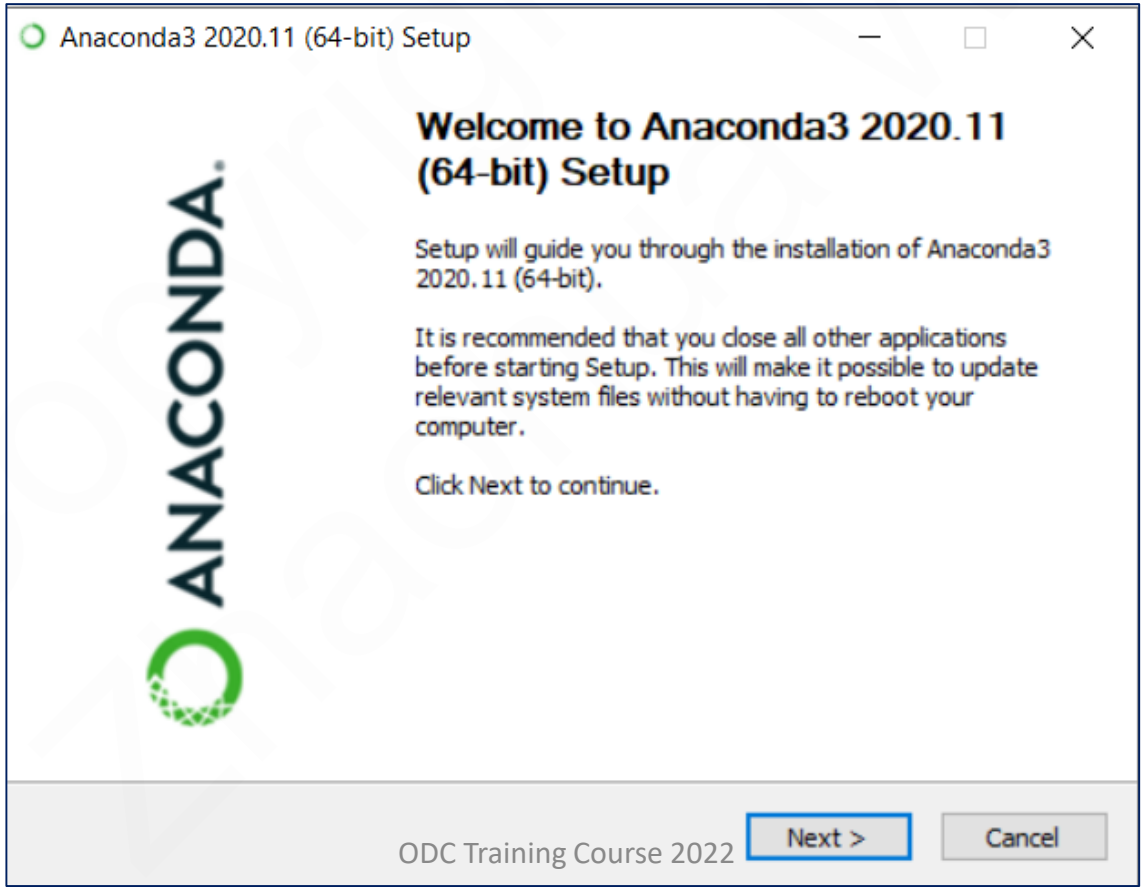
Anaconda Installers

Windows	MacOS	Linux
Python 3.9	Python 3.9	Python 3.9
64-Bit Graphical Installer (594 MB)	64-Bit Graphical Installer (591 MB)	64-Bit (x86) Installer (659 MB)
32-Bit Graphical Installer (488 MB)	64-Bit Command Line Installer (584 MB)	64-Bit (Power8 and Power9) Installer (367 MB)
	64-Bit (M1) Graphical Installer (316 MB)	64-Bit (AWS Graviton2 / ARM64) Installer (568 MB)
	64-Bit (M1) Command Line Installer (305 MB)	

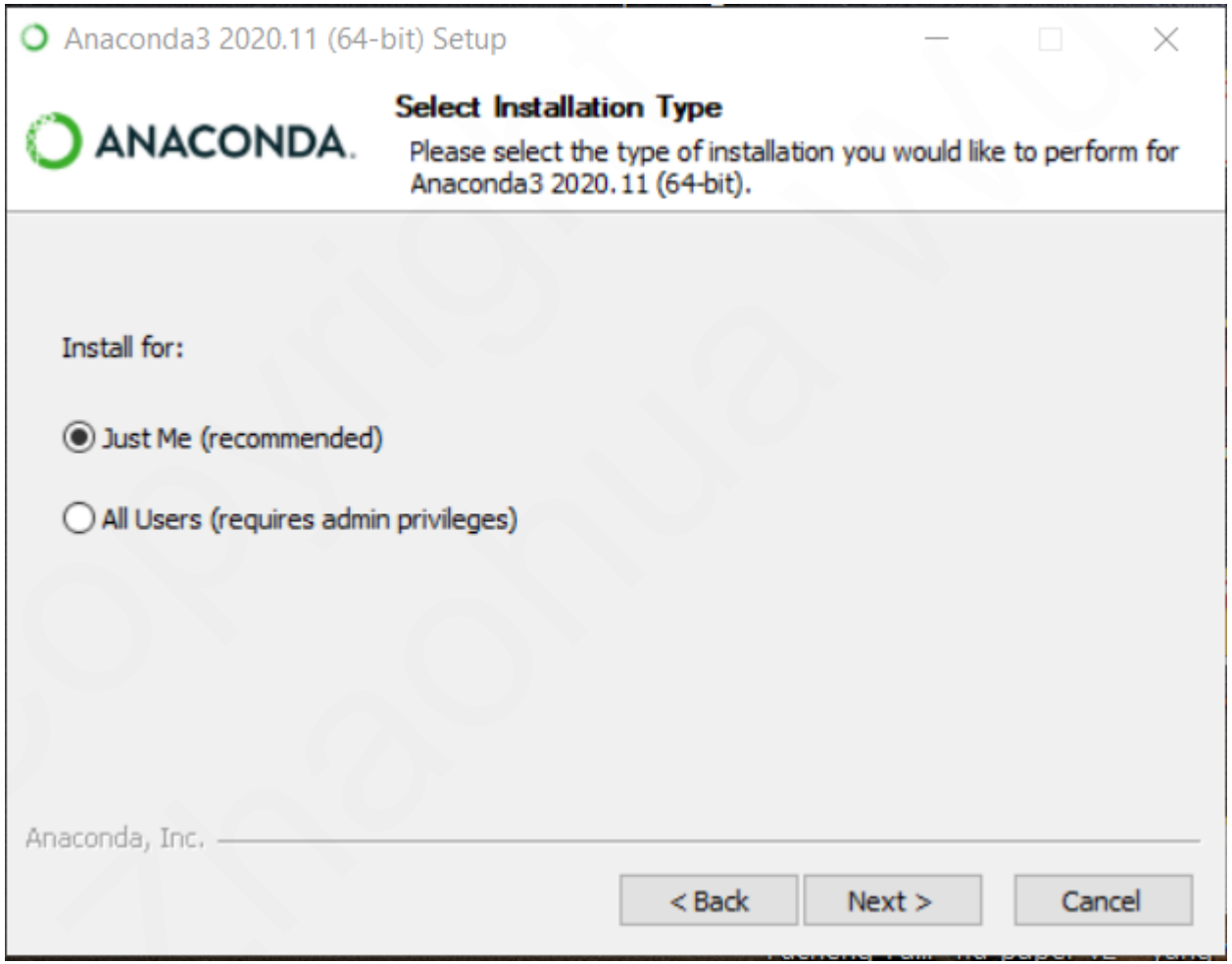


INSTALL


This PC
Downloads
EEMD_New
General Physics
Name
Date modified
Type
Size
Anaconda3-2020.11-Windows-x86_64
1/8/2021 9:47 AM
Application
468,161 KB


Anaconda3 2020.11 (64-bit) Setup
Welcome to Anaconda3 2020.11 (64-bit) Setup
ANACONDA.
Setup will guide you through the installation of Anaconda3 2020.11 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

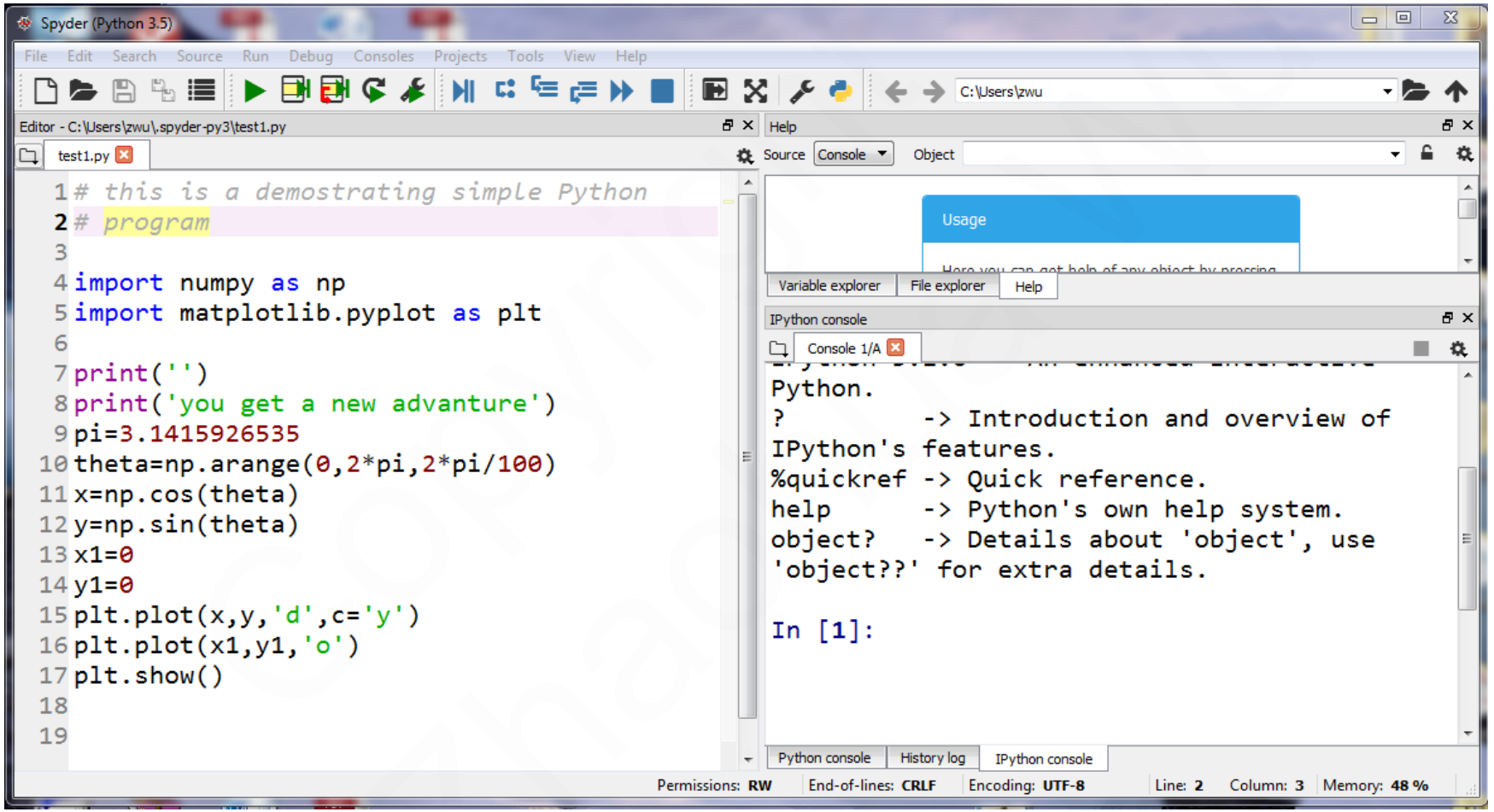
INSTALL


Anaconda3 2020.11 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2020.11 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel

START SPIDER (WINDOW)

- Click “Start”
- Click “All Program”
- Click “Anaconda3 (64 bits)”
- Scroll down and click “Spyder”

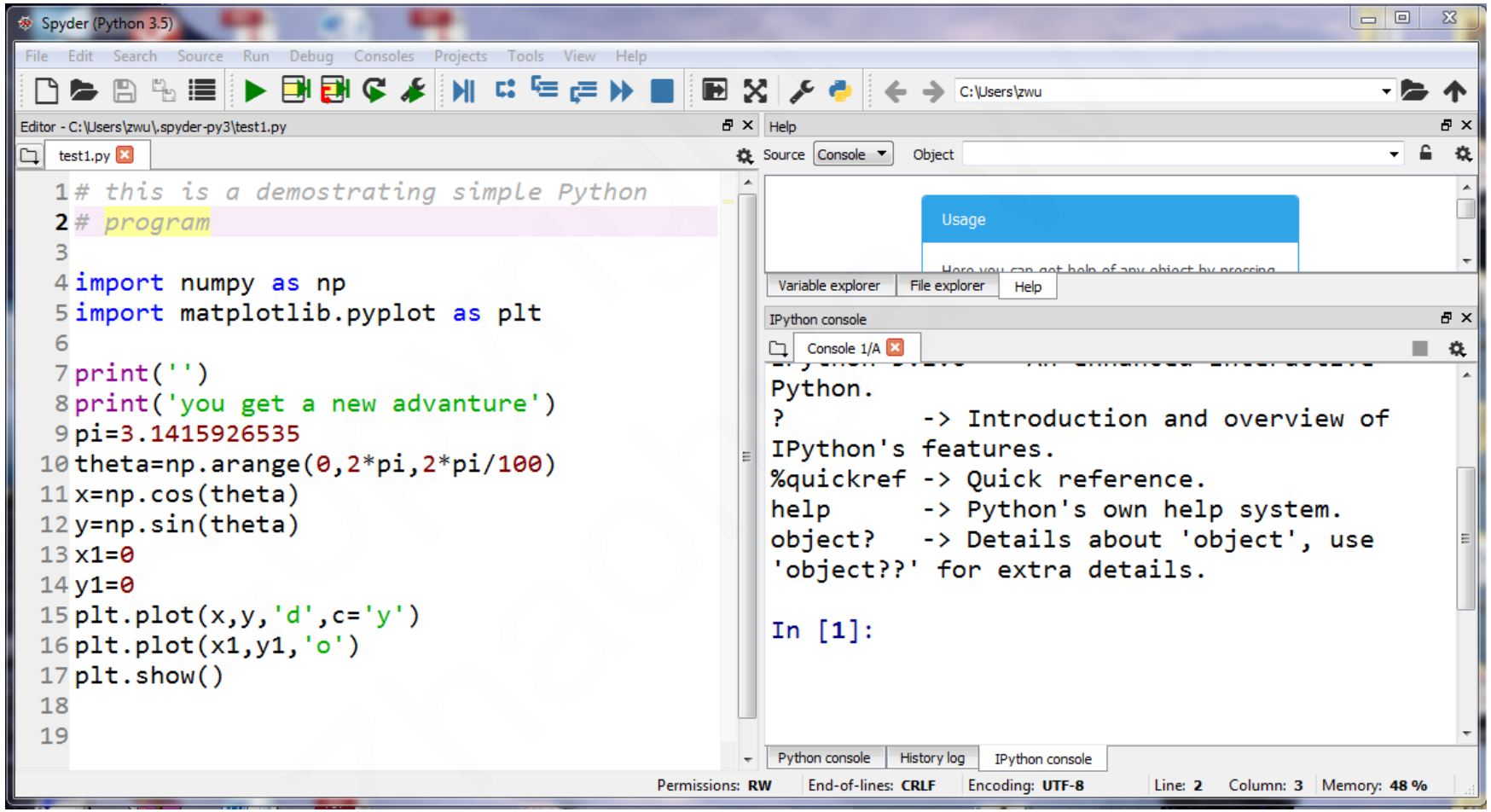
SPYDER USER INTERFACE



START SPIDER (OSX)

- Click “Go”
- Scroll down and click “Applications”; “Applications” window appears
- In the “Applications” window, double click “Anaconda-Navigator”, a new window of “Anaconda-Navigator” appears
- Click “launch” of “Spyder panel” (third panel)

SPYDER USER INTERFACE

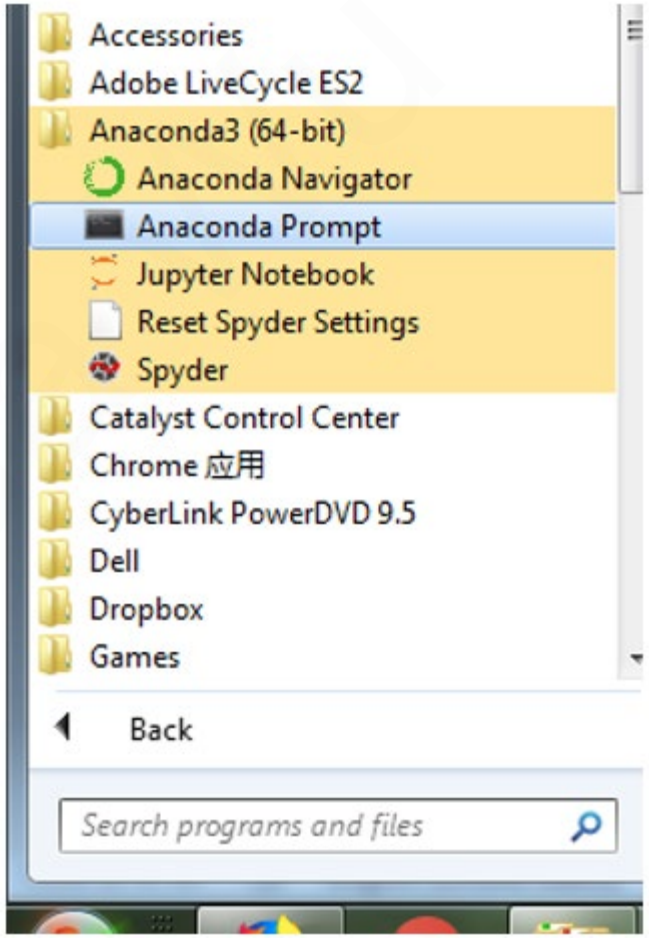


INSTALLING TOOLBOXES

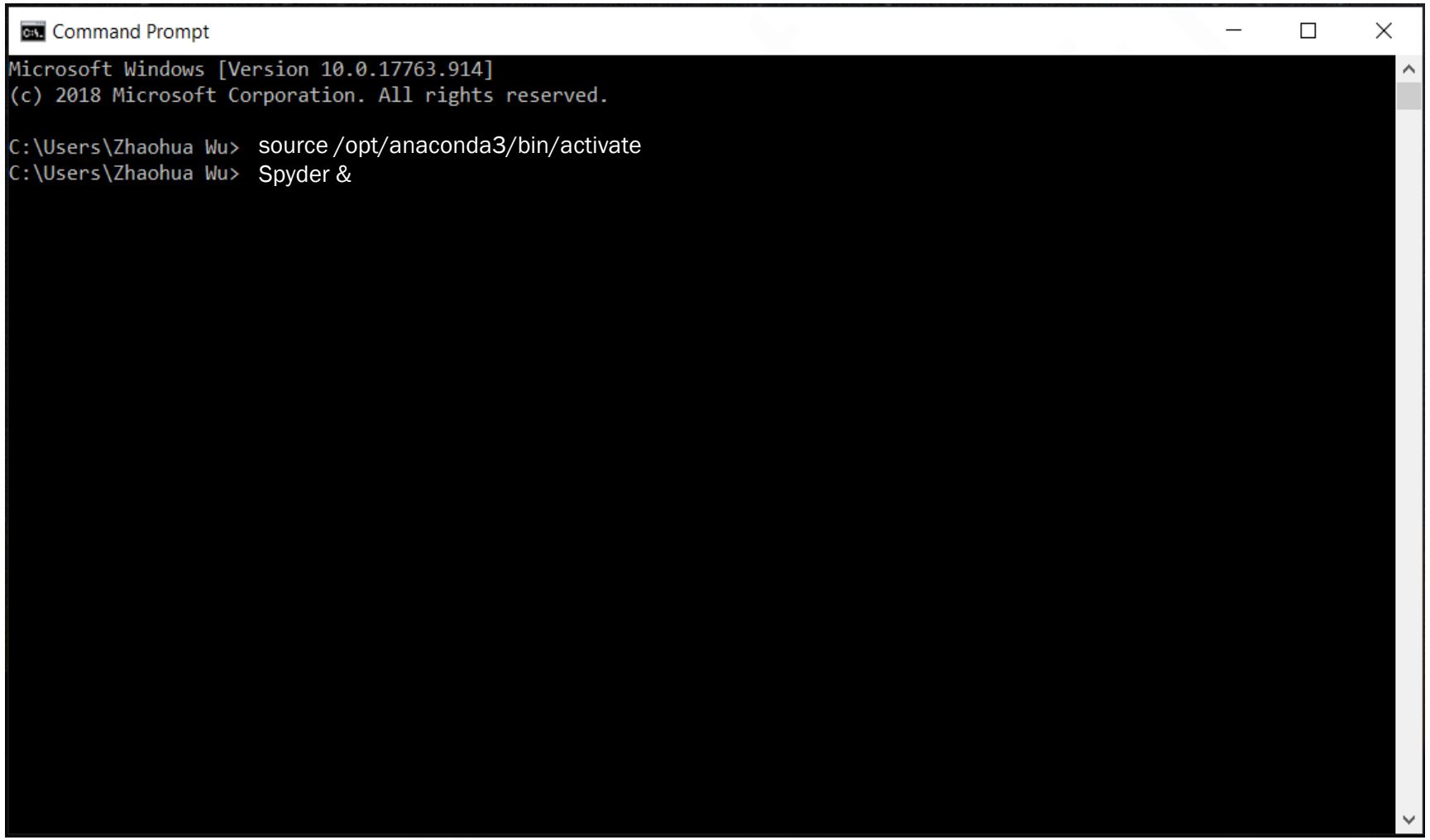
- For NETCDF4:
  - conda install -c anaconda netcdf4

- For Basemap:
  - conda install -c anaconda basemap
  - conda install -c conda-forge cartopy

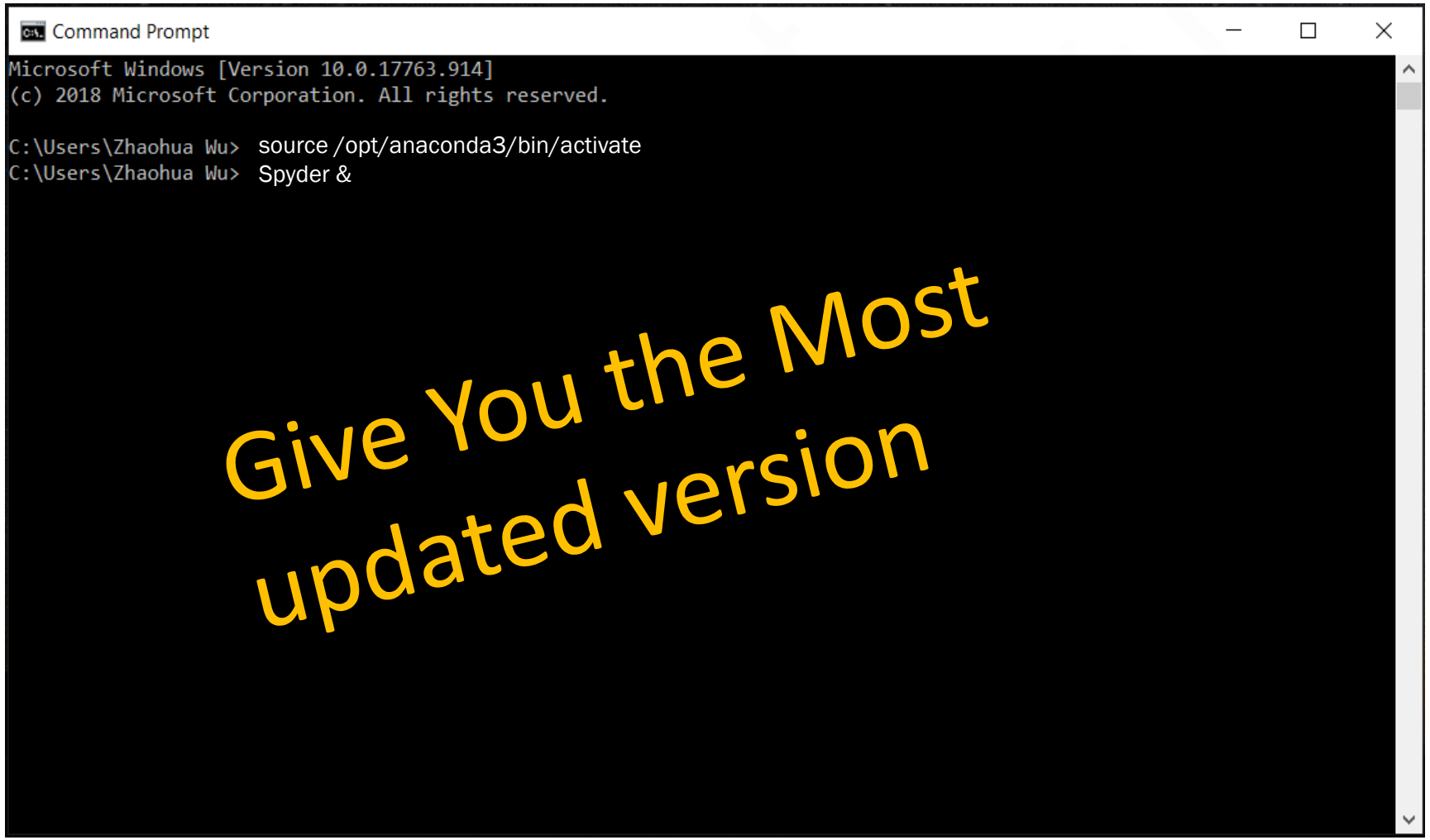
  - **Please use Anaconda Navigator to launch spyder.**



INSTALLING TOOLBOXES



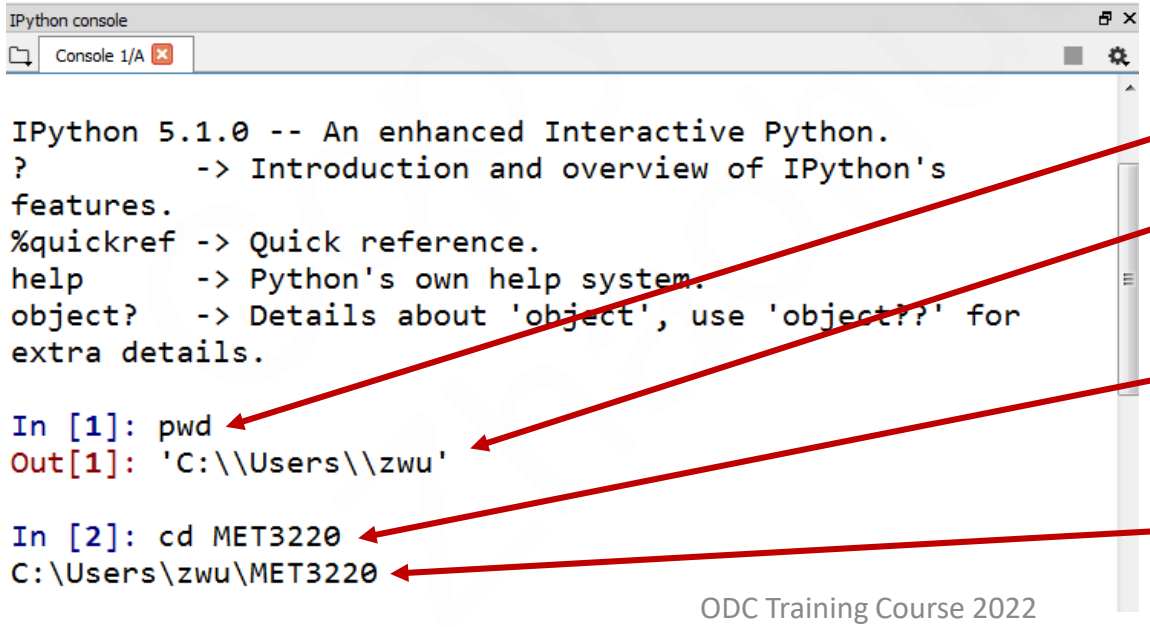
INSTALLING TOOLBOXES



LET'S HAVE FUN!

SETTING UP YOUR WORK DIRECTORY

- It will turn out more convenient to work in a directory for your learning
  - Whenever initiate you ipython, go to your working directory



1. Check the current directory with “pwd”

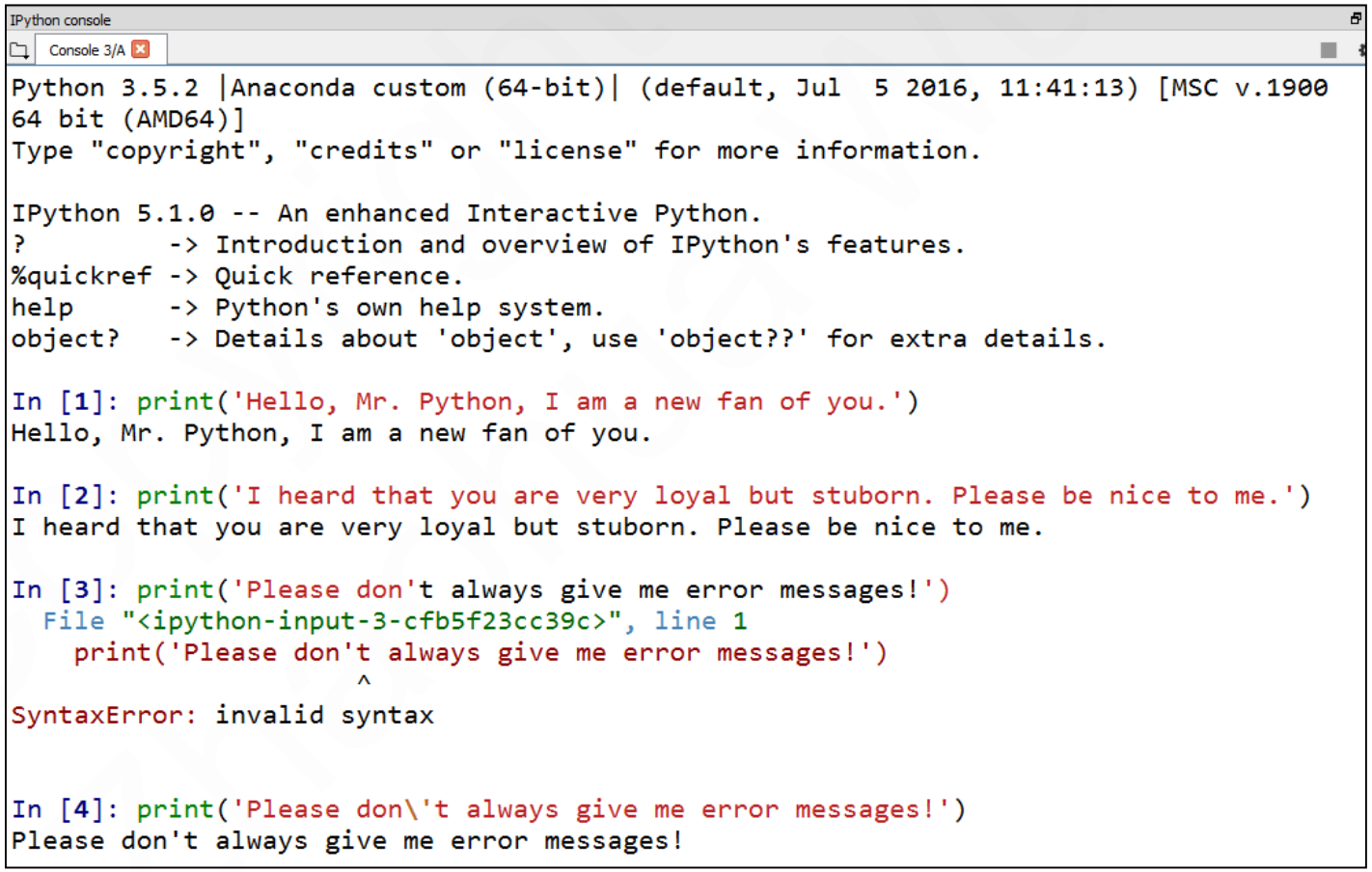
My home dir.

2. Go to your working directory with “cd”

My working dir.

COMMAND LINE INTERPRETATION

- IPython console



COMMAND LINE INTERPRETATION

- Advantages
  - Direct
  - Quick
- Disadvantages
  - Hardly workable for a real world job that contains numerous lines
  - slow

COLLECTION OF INSTRUCTIONS

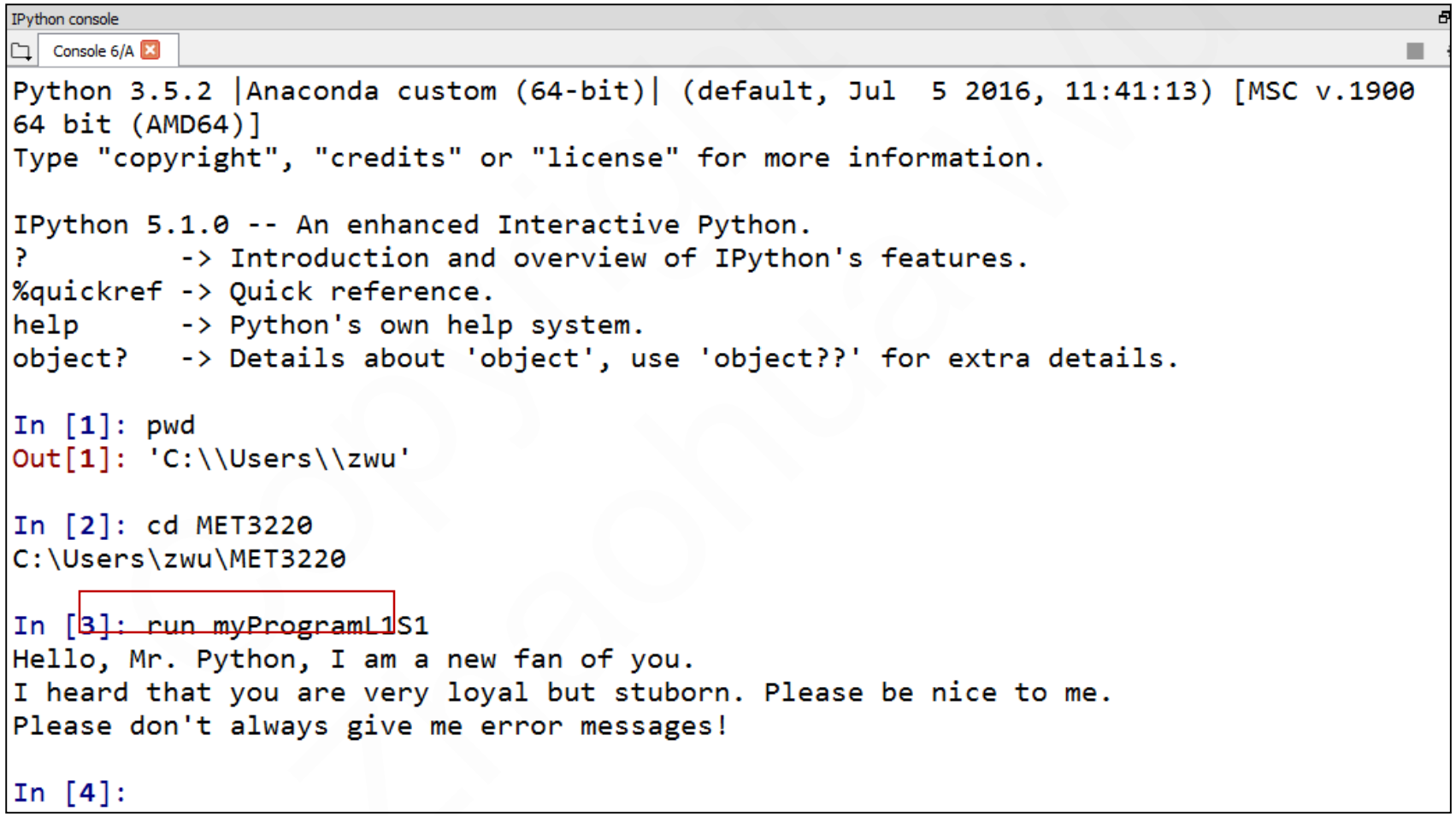
- Save in a file (e.g., myProgramL1S1.py)

Editor - C:\Users\zwu\MET3220\myProgramL1S1.py

myProgramL1S1.py

```
1 # this is a demostrating simple Python
2 # program
3
4 print('Hello, Mr. Python, I am a new fan of you.')
5 print('I heard that you are very loyal but stuborn. Please be nice
6 print('Please don\'t always give me error messages!')
7
```

RUNNING A COLLECTION OF INSTRUCTIONS



CAPABILITY OF PYTHON

IPython console

Console 7/A

```
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.1900
64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print('Mr. Python, are you a copy cat?')
Mr. Python, are you a copy cat?

In [2]: print('NO, I AM NOT')
NO, I AM NOT

In [3]: print('Show me')
Show me

In [4]: print('OK')
OK
```

ANOTHER EXAMPLE

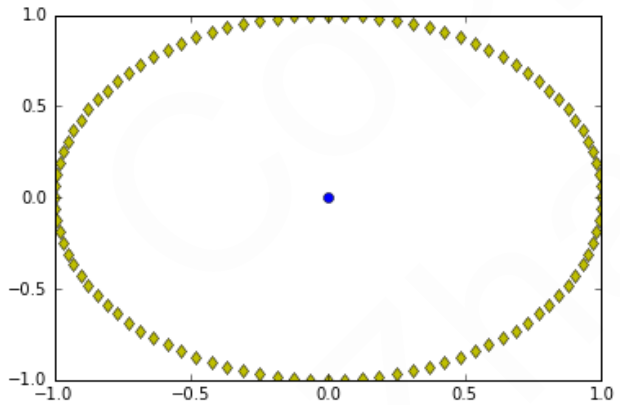
myProgramL1S1.py | myProgramL1S2.py

```
1 # this is a demostrating simple Python
2 # program for multi-tasks
3
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 print('')
8 print('you get a new advanture')
9 pi=3.1415926535
10 theta=np.arange(0,2*pi,2*pi/100)
11 x=np.cos(theta)
12 y=np.sin(theta)
13 x1=0
14 y1=0
15 plt.plot(x,y,'d',c='y')
16 plt.plot(x1,y1,'o')
17 plt.show()
18
19
```

- Type the codes, save it as a file with a name at a desired directory

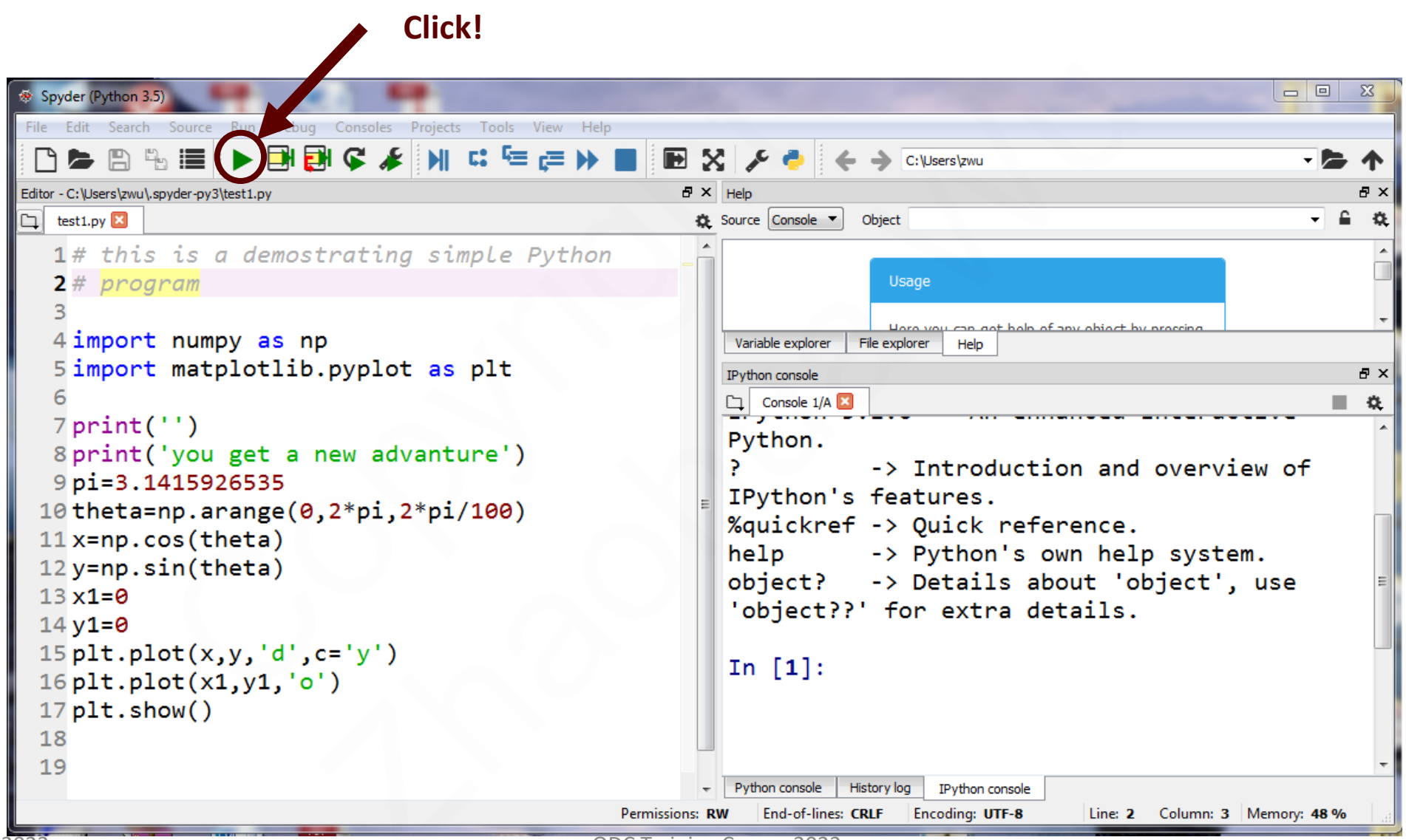
RESULTS

```
In [5]: run myProgramL1S2
ERROR:root:File `'myProgramL1S2.py'` not found.

In [6]: cd ..
C:\Users

In [7]: pwd
Out[7]: 'C:\\Users'

In [8]: cd zwu\MET3220
C:\Users\zwu\MET3220

In [9]: run myProgramL1S2

you get a new advanture
```




```
In [10]:
```

SPYDER USER INTERFACE



OUTLINE

- Computer and Modern Computing
- Preliminary Python Resources
- **Basic Python Scripting Language Structure**
- An Example

MATHEMATICAL OPERATORS

- Basic mathematical operators
  - addition “+”
  - subtraction “-”
  - multiplication “*”
  - division “/”
  - power “**”
- Priority (from high to low)
  - parenthesis
  - Power operation
  - Multiplicative operation
  - Additive operation

EXAMPLES: OPERATORS

Console 1/A

```
In [2]: var1=1

In [3]: var2=2

In [4]: var3=3

In [5]: var4=4

In [6]: var5=var1+var2*var3**var4

In [7]: var5
Out[7]: 163

In [8]: var6=(var1+var2)*var3**var4

In [9]: var6
Out[9]: 243

In [10]: var7=(var1+var2*var3)**var4

In [11]: var7
Out[11]: 2401

In [12]: var8=var1/var2*var3

In [13]: var9=var1/var3*var2
```

priority

Parenthesis matters

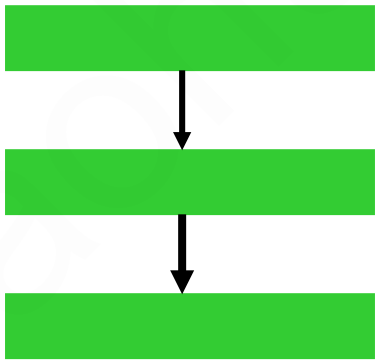
Left to Right

OTHER MATHEMATICAL OPERATORS

- Truncating division “ / /”
- Modulo operator “%”
- Logical operators
- Comparison operators
  - Equal to “==”
  - Not Equal to “ ! =”
  - Less than “<”
  - Greater than “>”
  - Less than or equal to “<=”
  - Greater than or equal to “>=”
- Chained comparison

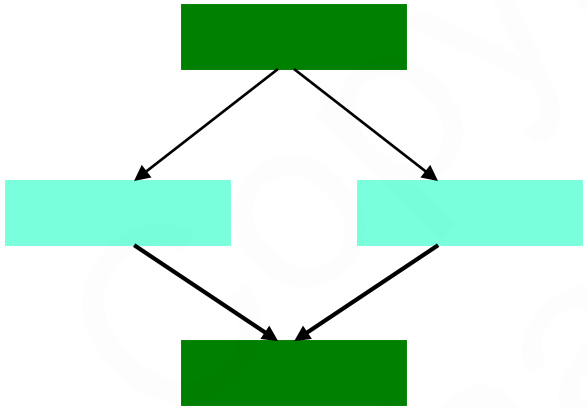
CONTROL FLOW

- “Control flow” is the order in which statements are executed
- Until now, control flow has been sequential – the next statement executed is the next one that appears, in order, in Python

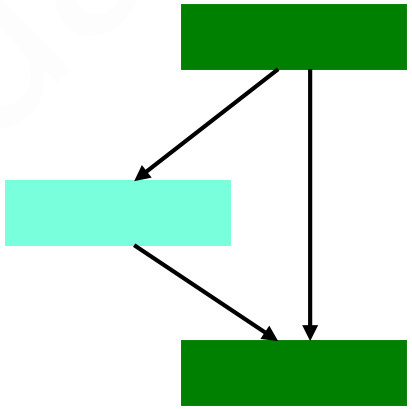


CONDITIONAL CONTROL FLOW

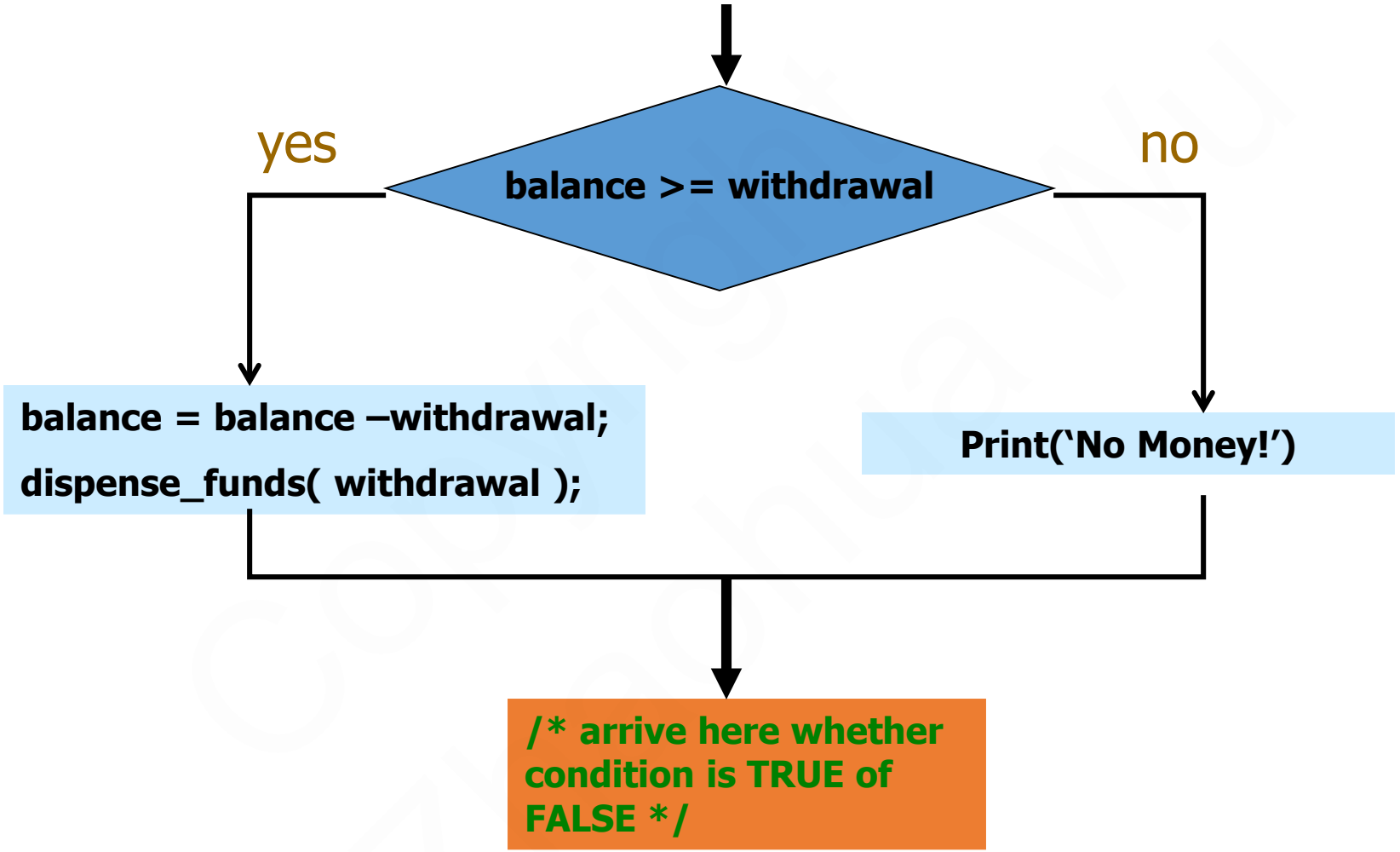
Choosing which of two (or more) statements to execute before continuing



Choosing whether or not to skip a statement before continuing

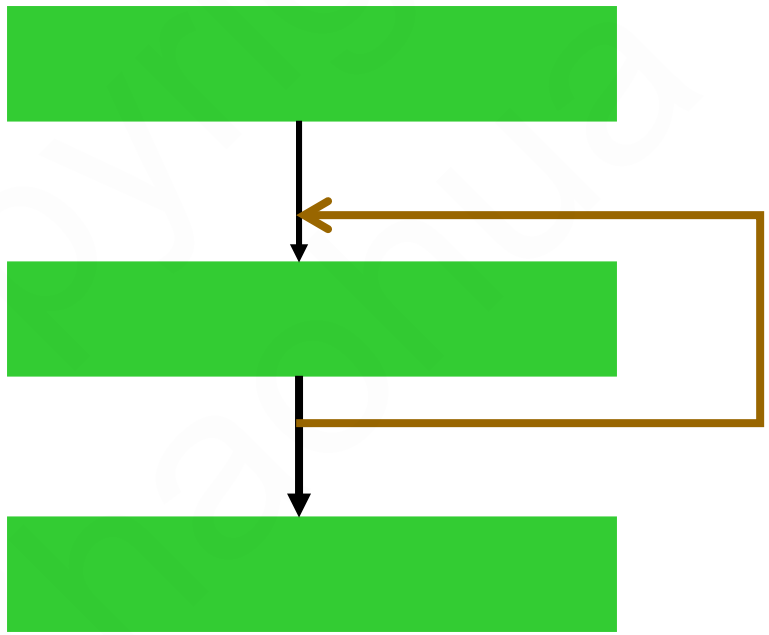


if-else CONTROL FLOW



A NEW TYPE OF CONTROL LOOP

- Sometimes we want to repeat a block of code. This is called a ***loop***



LOOPS TO ADD FOUR NUMBERS

myProgramL6S1.py | myProgramL6S3.py | myProgramL6S4.py | myProgramL6S5.py | myProgramL6S6.py

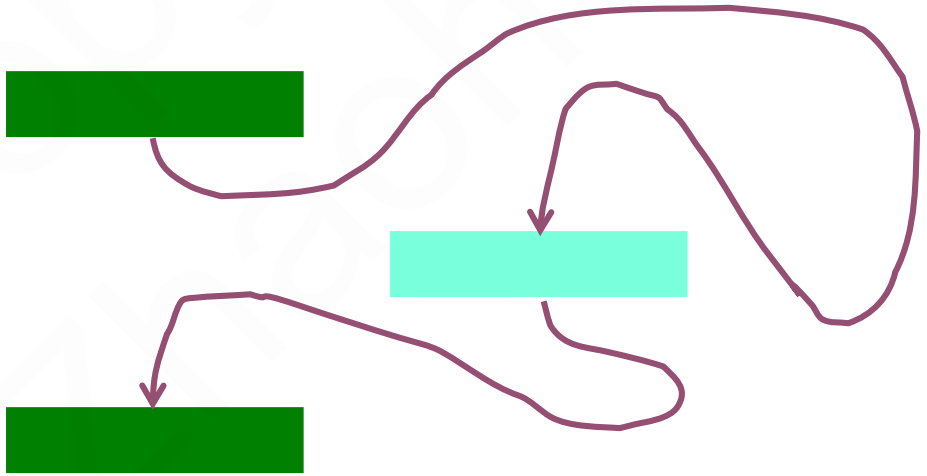
```
1  #  input 4 numbers and calculate their sum
2
3  sumOfVars = 0
4
5  var = float( input ('Please input a number: ') )
6  sumOfVars = sumOfVars + var
7
8  var = float( input ('Please input a number: ') )
9  sumOfVars = sumOfVars + var
10
11 var = float( input ('Please input a number: ') )
12 sumOfVars = sumOfVars + var
13
14 var = float( input ('Please input a number: ') )
15 sumOfVars = sumOfVars + var
16
17
18 print('The sum of four variables are: ', sumOfVars)
19
```

myProgramL6S6.py | myProgramL6S7.py

```
1  #  input 4 numbers and calculate their sum
2
3  sumOfVars = 0
4
5  for i in range(0,4):
6      var = float( input ('Please input a number: ') )
7      sumOfVars = sumOfVars + var
8
9  print('The sum of four variables are: ', sumOfVars)
10
```

ANOTHER FORM OF CONTROL FLOW

- "Functions" (or "procedures") allow you to "visit" a chunk of code and then come back
- The functions maybe elsewhere in your own program, or may be code in another file altogether



ONE (BIG) IDEA BEHIND FUNCTIONS

- Identify a “sub-problem” that has to be solved in your program
- Solve that sub-problem and write the code for it only once
- Give that code a name: that makes it a function
- Whenever you see that same sub-problem again, use the function name to say “go to that code now to take care of this problem, and don’t come back until you are done”

PREVIOUS FUNCTIONS WE DEFINED

holidayBush.py

```
"""
This is a program to printout holiday bush
An example of nested loop
"""

def holidaybush(n,char):
    z=n-1
    x=1
    for i in range(0,n):
        for i in range(0,z):
            print(' ',end='')
        for i in range(0,x):
            print(char,end='')
#        for i in range(0,z):
#            print(' ',end='')
        x=x+2
        z=z-1
        print()


holidaybush(5,'*')
holidaybush(10,'+')
```

Console 1/A

```
In [21]: run holidayBush.py
    *
   ***
  *****
 *******
*********
          +
         +++
        +++++
       +++++++
      +++++++++
     ++++++++++
    ++++++++++++
   ++++++++++++++
  ++++++++++++++++
 ++++++++++++++++++
++++++++++++++++++++
```

BUILDING A CAR

Question: How would you make a car?

BUILDING A CAR

Recipe 1: procedure-wise

1. Make a frame
2. Make a engine
3. Make four tires

…

Repeat the process to make another car

Assemble Factory

Recipe 2: assembling

1. Import necessary parts from different parts factory
2. Assemble them

Repeat the process to make another car

OBJECTED ORIENTED

Assemble Factory

Part A
(Supply Factory A)

Part B
(Supply Factory B)

Part C
(Supply Factory C)

Import the necessary parts only when one needs them to more effectively assemble a car.

PYTHON STRUCTURE

Math
(math functions)

numpy
(math function and more)

matplotlib
(plotting functions)

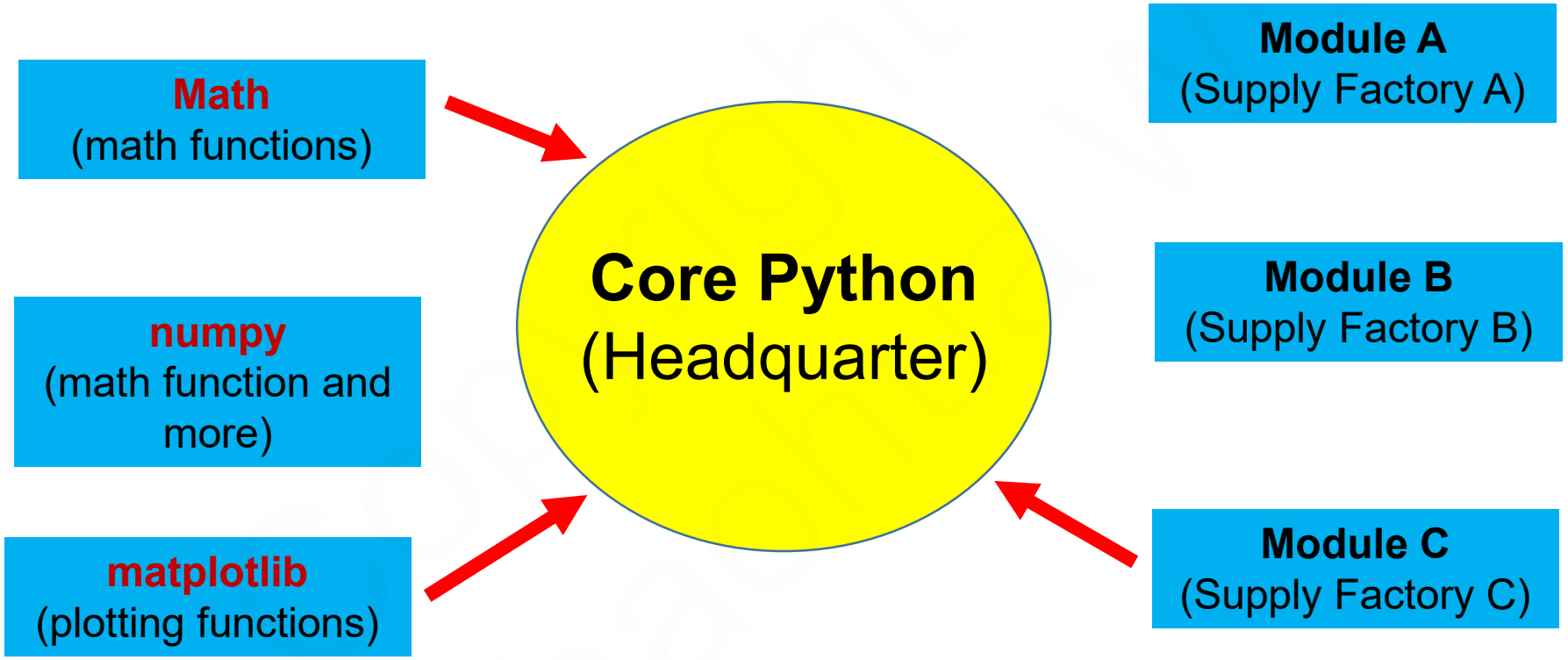
Core Python
(Headquarter)

Module A
(Supply Factory A)

Module B
(Supply Factory B)

Module C
(Supply Factory C)

PYTHON STRUCTURE

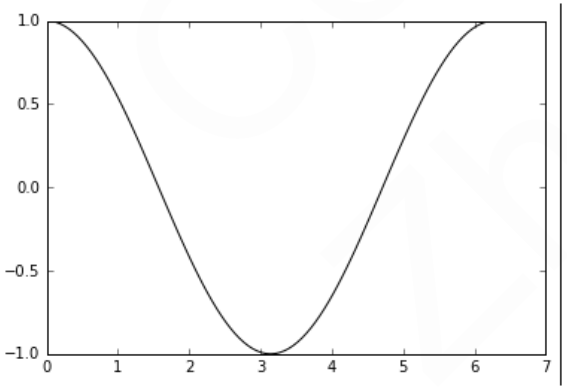


Import the necessary modules only when one needs them to more effectively use computational system

TEST MATH FUNCTIONS

basemap_netcdf.py | myProgramL5S2.py | myProgramL5S3.py | myProgramL5S4.py | myProgramL5S5.py

```
1 # plot a circle with a read-in center coordiates
2 # and a read-in radius
3
4
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 theta = np.arange(0,2*np.pi,2*np.pi/100)
9 x = np.cos(theta)
10
11 plt.plot(theta,x,c='k')
12 plt.show()
```



Just Do It!

OUTLINE

- Computer and Modern Computing
- Preliminary Python Resources
- Basic Python Scripting Language Structure
- **An Example**

GMSTA.DAT IN NOTEPAD

gmsta - Notepad

File Edit Format View Help

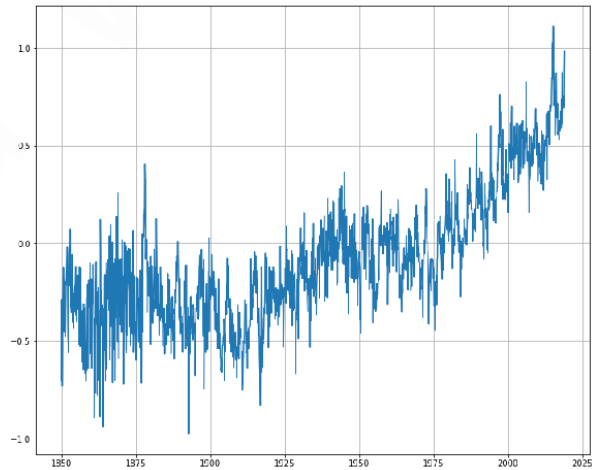
```
1850 -0.700 -0.286 -0.732 -0.563 -0.327 -0.213 -0.125 -0.237 -0.439 -0.451 -0.187 -0.257
1850     22     20     19     19     19     21     22     23     24     23     24     25
1851 -0.296 -0.356 -0.479 -0.441 -0.295 -0.197 -0.212 -0.157 -0.101 -0.057 -0.020 -0.051
1851     23     22     20     21     20     21     22     23     19     20     18     20
1852 -0.315 -0.477 -0.502 -0.557 -0.211 -0.040 -0.018 -0.202 -0.125 -0.216 -0.193  0.073
1852     23     22     22     23     23     24     23     24     21     22     22     25
1853 -0.182 -0.327 -0.309 -0.355 -0.268 -0.175 -0.059 -0.148 -0.404 -0.362 -0.255 -0.437
1853     25     26     26     24     25     27     26     29     28     29     26     27
1854 -0.365 -0.282 -0.286 -0.353 -0.233 -0.219 -0.227 -0.167 -0.119 -0.192 -0.367 -0.233
1854     30     29     29     28     28     27     28     32     28     30     29     28
1855 -0.169 -0.401 -0.306 -0.223 -0.338 -0.167 -0.271 -0.163 -0.336 -0.217 -0.214 -0.518
1855     29     30     27     27     29     28     30     30     28     25     25     23
1856 -0.122 -0.376 -0.528 -0.380 -0.129 -0.291 -0.304 -0.309 -0.461 -0.391 -0.608 -0.449
1856     25     27     25     26     27     26     25     27     26     28     31     26
1857 -0.513 -0.345 -0.434 -0.649 -0.570 -0.310 -0.541 -0.329 -0.393 -0.474 -0.668 -0.358
1857     26     26     26     25     26     24     27     28     26     26     25     24
1858 -0.528 -0.708 -0.548 -0.512 -0.653 -0.582 -0.329 -0.278 -0.332 -0.211 -0.643 -0.304
1858     26     26     27     26     25     23     23     23     23     21     22     25
1859 -0.308 -0.189 -0.327 -0.195 -0.307 -0.248 -0.274 -0.100 -0.565 -0.249 -0.315 -0.363
1859     23     23     22     20     21     19     22     22     20     21     19     22
1860 -0.181 -0.426 -0.642 -0.332 -0.293 -0.309 -0.109 -0.182 -0.217 -0.195 -0.515 -0.762
1860     24     20     21     21     20     18     19     20     18     18     18     17
1861 -0.893 -0.508 -0.464 -0.386 -0.768 -0.189 -0.221 -0.094 -0.330 -0.362 -0.417 -0.247
1861     18     16     14     13     12     13     14     15     15     16     16     16
1862 -0.742 -0.780 -0.403 -0.240 -0.225 -0.336 -0.344 -0.700 -0.420 -0.420 -0.766 -0.887
1862     15     14     15     15     17     17     17     16     17     16     14     13
```

year and monthly values

SELECT PARTICULAR PORTION OF DATA

Jessica_Pendulum5.py | Jessica_Pendulum.py | coronavirus simulation_version2.py | readNumericArrayFile.py

```
"""
This is a piece of code to read in a text file with different types of
entries separated by commas.
"""

import numpy as np
import matplotlib.pyplot as plt

gmsta=np.loadtxt('gmsta.txt')

numYears=round(len(gmsta[:,0])/2)

temperature2D=gmsta[0:342:2,1:13]
coverage2D=gmsta[1:342:2,1:13]

numMonths=numYears*12
temperature1D=np.reshape(temperature2D, numMonths, order='C')
coverage1D=np.reshape(coverage2D, numMonths, order='C')

# test the wrong reshape
#temperature1Dv2=np.reshape(temperature2D, numMonths, order='F')
#coverage1Dv2=np.reshape(coverage2D, numMonths, order='F')

t=np.linspace(1850.0416667,2019.9583333,numMonths)
plt.plot(t[0:numMonths-11],temperature1D[0:numMonths-11])
plt.grid()
plt.gcf().set_size_inches(12,10)
plt.show()
```

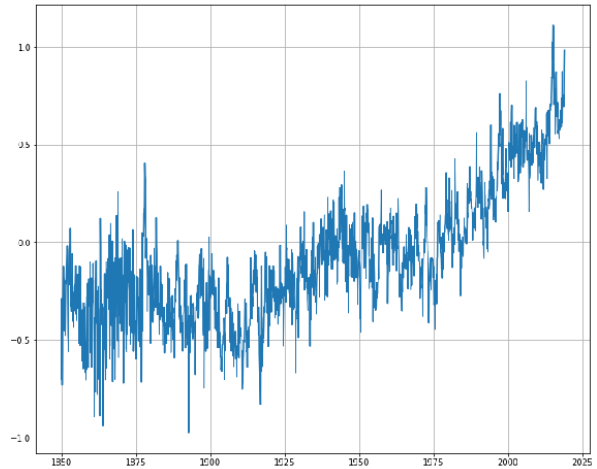


SELECT PARTICULAR PORTION OF DATA

Jessica_Pendulum5.py | Jessica_Pendulum.py | coronavirus simulation_version2.py | readNumericArrayFile.py

```
"""
This is a piece of code to read in a text file with different types of
entries separated by commas.
"""

import numpy as np
import matplotlib.pyplot as plt

gmsta=np.loadtxt('gmsta.txt')

numYears=round(len(gmsta[:,0])/2)

temperature2D=gmsta[0:342:2,1:13]
coverage2D=gmsta[1:342:2,1:13]

numMonths=numYears*12
temperature1D=np.reshape(temperature2D, numMonths, order='C')
coverage1D=np.reshape(coverage2D, numMonths, order='C')

# test the wrong reshape
#temperature1Dv2=np.reshape(temperature2D, numMonths, order='F')
#coverage1Dv2=np.reshape(coverage2D, numMonths, order='F')

t=np.linspace(1850.0416667,2019.9583333,numMonths)
plt.plot(t[0:numMonths-11],temperature1D[0:numMonths-11])
plt.grid()
plt.gcf().set_size_inches(12,10)
plt.show()
```

From 2D to 1D



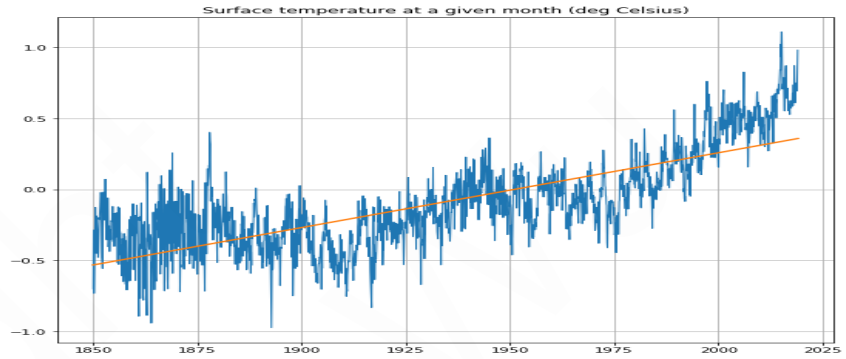
FROM GMSTA TO ITS TREND

$$T_i = a_0 + a_1 t + \varepsilon_i$$

- In this case:
  - x is the time axis (**t**, **timeVar**), starting from January 1850, ending at January at January 2020, a total of 2041 data points
  - y (**T**) is the corresponding **"gmsta"** data, starting from January 1850, ending at January at January 2020, a total of 2041 data points.
- Calculations

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$




```
gmsta=np.loadtxt('gmsta.dat')
numRows=len(gmsta[:,0])
numYears=round(numRows/2)

temperature2D=gmsta[0:numRows:2,1:13]
coverage2D=gmsta[1:numRows:2,1:13]

numMonths=numYears*12
tGMSTA=np.linspace(1850.0416667,2019.9583333,numMonths)

temperature1D=np.reshape(temperature2D, numMonths, order='C')
coverage1D=np.reshape(coverage2D, numMonths, order='C')
```

```
# Part II: trend of a gmsta
timeVar=tGMSTA[0:numMonths-11]
values=temperature1D[0:numMonths-11]
time,trendGMSTA=stat.LinearTrend(timeVar, values)

### Example I:  Displaying gmsta.
#---------------------------------------------------------------
# cerat figure, axes instances
fig1=plt.figure()
plt.plot(tGMSTA[0:numMonths-11], temperature1D[0:numMonths-11], \
         linewidth = 1 )
plt.plot(time,trendGMSTA,linewidth=2)
plt.grid()
plt.title('Surface temperature at a given month (deg Celsius)')
plt.gcf().set_size_inches(9,9)
plt.show()
##===============================================================
```

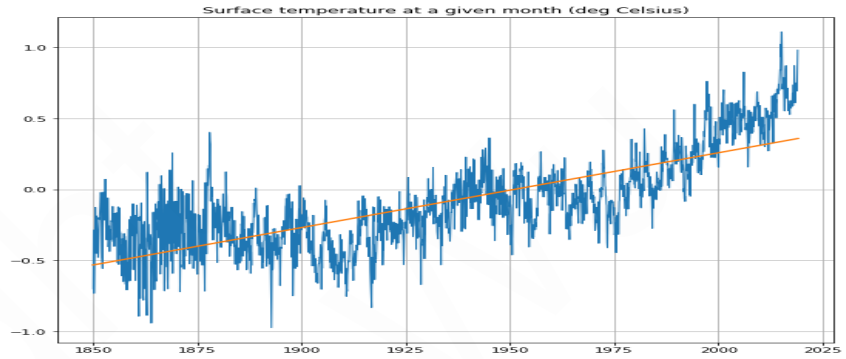
FROM GMSTA TO ITS TREND

$$T_i = a_0 + a_1 t + \varepsilon_i$$

- In this case:
  - x is the time axis (**t**, **timeVar**), starting from January 1850, ending at January at January 2020, a total of 2041 data points
  - y (**T**) is the corresponding **"gmsta"** data, starting from January 1850, ending at January at January 2020, a total of 2041 data points.
- Calculations

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$




```
gmsta=np.loadtxt('gmsta.dat')
numRows=len(gmsta[:,0])
numYears=round(numRows/2)

temperature2D=gmsta[0:numRows:2,1:13]
coverage2D=gmsta[1:numRows:2,1:13]

numMonths=numYears*12
tGMSTA=np.linspace(1850.0416667,2019.9583333,numMonths)

temperature1D=np.reshape(temperature2D, numMonths, order='C')
coverage1D=np.reshape(coverage2D, numMonths, order='C')
```

```
# Part II: trend of a gmsta
timeVar=tGMSTA[0:numMonths-11]
values=temperature1D[0:numMonths-11]
time,trendGMSTA=stat.LinearTrend(timeVar, values)

### Example I:  Displaying gmsta.
#---------------------------------------------------------------
# cerat figure, axes instances
fig1=plt.figure()
plt.plot(tGMSTA[0:numMonths-11], temperature1D[0:numMonths-11], \
         linewidth = 1 )
plt.plot(time,trendGMSTA,linewidth=2)
plt.grid()
plt.title('Surface temperature at a given month (deg Celsius)')
plt.gcf().set_size_inches(9,9)
plt.show()
##===============================================================
```

STAT.LINEARTREND(…)

```
 7 import numpy as np
 8 import numpy.ma as ma
 9 import matplotlib.pyplot as plt
10 import MyStatistics2020 as stat
11 from mpl_toolkits.basemap import Basemap
12 from netCDF4 import Dataset
```

```
18 gmsta=np.loadtxt('gmsta.dat')
19 numRows=len(gmsta[:,0])
20 numYears=round(numRows/2)
21
22 temperature2D=gmsta[0:numRows:2,1:13]
23 coverage2D=gmsta[1:numRows:2,1:13]
24
25 numMonths=numYears*12
26 tGMSTA=np.linspace(1850.0416667,2019.9583333,numMonths)
27
28 temperature1D=np.reshape(temperature2D, numMonths, order='C')
29 coverage1D=np.reshape(coverage2D, numMonths, order='C')
```

```
38 # Part II: trend of a gmsta
39 timeVar=tGMSTA[0:numMonths-11]
40 values=temperature1D[0:numMonths-11]
41 time,trendGMSTA=stat.LinearTrend(timeVar, values)
42
43 ### Example I:  Displaying gmsta.
44 #-------------------------------------------------------------
45 # cerat figure, axes instances
46 fig1=plt.figure()
47 plt.plot(tGMSTA[0:numMonths-11], temperature1D[0:numMonths-11], \
48          linewidth = 1 )
49 plt.plot(time,trendGMSTA,linewidth=2)
50 plt.grid()
51 plt.title('Surface temperature at a given month (deg Celsius)')
52 plt.gcf().set_size_inches(9,9)
53 plt.show()
54 ##=================================================================
```

A analysis package with a name "**MyStatistics2020.py**" created by you and saved in the same directory as your final project driver.

Final_Project_2020.py | MyStatistics2020.py*

```
107 # Linear trend
108 def LinearTrend(timeVar, values):
109     xp = np.zeros([len(timeVar)])
110     yp = np.zeros([len(timeVar)])
111     valueLength=len(timeVar)
112     sumLength=0
113     for i in range(len(timeVar)):
114         if (values[i] > -1e+30 ):
115             xp[sumLength] = timeVar[i]
116             yp[sumLength] = values[i]
117             sumLength=sumLength+1
118
119     if sumLength > 0.5*valueLength:
120         x = xp[0:sumLength]
121         y = yp[0:sumLength]
122         meanX = Mean(x)
123         meanY = Mean(values)
124         varX,stdX = VarStd (x)
125         covXY = Covariance(x, y)
126
127         b1 = covXY / varX
128         b0 = meanY - b1*meanX
129 #         print('b0= ',b0,  '  b1= ',b1)
130
131         yReg=b0 + b1 * x
132
133         return x, yReg
134     else:
135         return np.array([-1e+30]),np.array([-1e+30])
136
```

MyStatistics2020.py

Final_Project_2020.py | MyStatistics2020.py*

```
"""
This package of code contains statistical functions created by the instructor
of MET3220, Zhaohua Wu

These functions are used for the completion of the final project of MET3220.

April, 2020

"""

import numpy as np


def Mean(values):
    sumValues=0
    sumLength=0
    valueLength = len(values)
    for i in range(valueLength):
        if (values[i] > -1e+30):
            sumValues=sumValues+values[i]
            sumLength=sumLength+1
    if sumLength > 0.5*valueLength:
        return sumValues/sumLength
    else:
        return -1e+30


```

$$a_0 = \overline{y} - a_1 \overline{x}$$

$$a_1 = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}$$

MyStatistics2020.py

Final_Project_2020.py | MyStatistics2020.py*

```
28 # Calculate the variance and standard deviation of an array
29 def varStd(values):
30     meanValue=Mean(values)
31     if meanValue > -1e+30:
32         sumValues=0
33         sumLength=0
34
35         for i in range(len(values)):
36             if (values[i] > -1e+30):
37                 diff=values[i]-meanValue
38                 sumValues=sumValues+diff*diff
39                 sumLength=sumLength+1
40
41         var=sumValues/sumLength
42         stdv=np.sqrt(var)
43
44         return var, stdv
45
46     else:
47         return -1e+30, -1e+30
48
49
50
```

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$

MyStatistics2020.py

Final_Project_2020.py | MyStatistics2020.py*

```
51 # Calculate covariance between x and y
52 def Covariance (x, y):
53     xp = np.zeros([len(x)])
54     yp = np.zeros([len(y)])
55     sumValues = 0
56     sumLength = 0
57     for i in range (len(x)):
58         if x[i]>-1e30 and y[i]>-1e30:
59             xp[sumLength]=x[i]
60             yp[sumLength]=y[i]
61             sumLength = sumLength +1
62     newX=xp[0:sumLength]
63     newY=yp[0:sumLength]
64     if sumLength > 0.5 * len(x):
65         meanX = Mean(newX)
66         meanY = Mean(newY)
67         for i in range (len(newX)):
68             diffx = newX[i] - meanX
69             diffy = newY[i] - meanY
70             sumValues =  sumValues + diffx * diffy
71         covar= sumValues/len(newX)
72         return covar
73     else:
74         return -1e30
75
76
```

Notice: In this function we have two variables

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$

MyStatistics2020.py

Final_Project_2020.py | MyStatistics2020.py*

```
# Calculate regression coefficients
def RegCoefs(xValues, yValues):
    xp = np.zeros([len(xValues)])
    yp = np.zeros([len(xValues)])
    valueLength = len(xValues)
    sumLength=0
    for i in range(len(xValues)):
        if (xValues[i] > -1e+30 and yValues[i] > -1e+30):
            xp[sumLength]= xValues[i]
            yp[sumLength]= yValues[i]
            sumLength=sumLength+1

    if sumLength > 0.5*valueLength:
        x =  xp[0:sumLength]
        y =  yp[0:sumLength]

        meanX = Mean(x)
        meanY = Mean(y)
        varX,stdX = VarStd (x)
        covXY = Covariance(x, y)

        b1 = covXY / varX
        b0 = meanY - b1*meanX

        return b0, b1
    else:
        return -1e+30, -1e+30
```

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$

MyStatistics2020.py

Final_Project_2020.py | MyStatistics2020.py*

```
107 # Linear trend
108 def LinearTrend(timeVar, values):
109     xp = np.zeros([len(timeVar)])
110     yp = np.zeros([len(timeVar)])
111     valueLength=len(timeVar)
112     sumLength=0
113     for i in range(len(timeVar)):
114         if (values[i] > -1e+30 ):
115             xp[sumLength] = timeVar[i]
116             yp[sumLength] = values[i]
117             sumLength=sumLength+1
118
119     if sumLength > 0.5*valueLength:
120         x = xp[0:sumLength]
121         y = yp[0:sumLength]
122         meanX = Mean(x)
123         meanY = Mean(values)
124         varX,stdX = VarStd (x)
125         covXY = Covariance(x, y)
126
127         b1 = covXY / varX
128         b0 = meanY - b1*meanX
129 #        print('b0= ',b0, '  b1= ',b1)
130
131         yReg=b0 + b1 * x
132
133         return x, yReg
134     else:
135         return np.array([-1e+30]),np.array([-1e+30])
136
```

$$a_0 = \bar{y} - a_1 \bar{x}$$

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$