ÖZYEĞİN UNIVERSITY

EE393 Python for Engineers

Dr. Orhan Gökçöl

orhan.gokcol@ozyegin.edu.tr

online

WEEK #1

2020-2021 Fall Semester

1

Welcome to EE393 Python for Engineers

- Who am I?
  - See my LinkedIn Profile
- Have you heard about Python?
- (Why) do we need to learn (another) programming language?

LETS GET TO
KNOW EACH
OTHER
LEARNINGSTUDIO

2

Orhan Gökçöl

- PhD in Aeronautics (İTÜ)
  - Computational modeling, visualization and data analysis for industrial problems
- Actively working in information security, cyber security, IT services & management, resilience and business continuity areas
  - Consultancy, training and Auditing – TÜV NORD (Germany)
- Faculty @Bahçeşehir Univ., School of Education (formerly @Mechatronics Engineering till 2015).
- Experience in software security and software management
- Did and managed many software projects in different fields including automotive, electronics, e-commerce, finance and education
- Been a software developer since 1987. Did development in various languages and frameworks including C/C++, Java, PHP, Python, R, Fortran, Rexx, System 360 Assembly, Pascal, Systems programming etc.

IMPORTANT!!!!

- You are **required** to have your mobile computer (Mac, Win or Linux are OK!) with you!!!.
- Alternatively:
  - For iPad, you may use “Pythonista” (your own responsibility, there will be no support ☺. You are on your own, but it seems very promising)
  - For android, I have no experience
  - For any internet-connected device (with a physical keyboard!), it is possible to use cloud-based Python (colab, azure notebooks, repl…)

EE393 aims at

- Make you comfortable in using Python to solve problems occurred in your engineering disciplines.
- Review important engineering libraries such as numpy, scipy, pandas and matplotlib.
- Semester mini projects -using python in engineering problem solving.

5

First things first….

- EE393 is a **programming course**
- It is not an “Introduction to….. “ course. You are assumed to have already passed CS101 or equivalent, thus, must feel yourself (a little bit) comfortable with a high level language (C, Java, C#, PHP, Ruby, JavaScript etc. all fine)
  - You don’t need to be an expert, but know the basic programming ideas such as **variables**, **functions**, **operators**, **loops**, **if**-s, programming logic in general
  - Otherwise, drop the course ☹

6

Course Logistics

- **Weekly Schedule**
  - **MONDAY, 08:40 (via Zoom Meetings)**
- **Course web support and distance learning**
  - I’ll use OzU LMS
- **Course e-mail (use this email for communication)**
  - ***orhan.gokcol@ozyegin.edu.tr***
- **Communication with course instructor and TAs**
  - We’ll have an active communication channel through LMS

✉ **(email / instant messaging) gokcol@gmail.com**

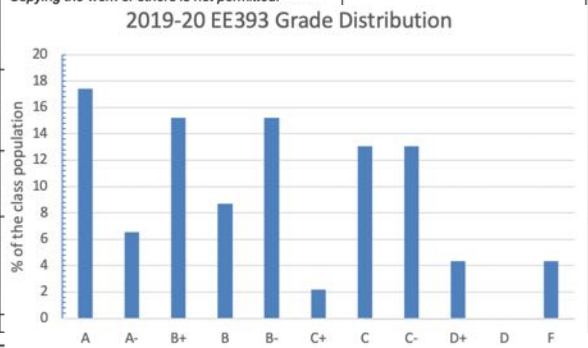
☎ **(business phone) (532)483-4545**

7

ASSESSMENT METHODS, WEIGHTS AND RULES

Type	Weight	Assessment Method	Implementation Rule	Makeup Rule
Final Exam	*25*	*Online with Respondus proctoring*	*Final exam will be closed books and notes. No calculation or communication devices will be allowed during the exam.*	BÜT exam will serve as the makeup for the final exam
Midterm Exam	*20*	*Online with Respondus proctoring*	*Midterm exam will be closed books and notes. No calculation or communication devices will be allowed during the exam.*	*Only valid excuses with an official report are accepted to qualify for a midterm makeup. At most one makeup will be given in the course due to health reports.*
Quiz	*5*	*Offline*	*Take-home quiz. Students will have a limited amount of time (e.g. 6 hours) to complete a task. Copying the work of others is not permitted.*	---
Homework	*25*	*Offline*		
Project	*15*	*Offline*		
Other	*10*	*Online collaboration - such as using LMS discussion boards effectively*		
Total	**100%**			

2019-20 EE393 Grade Distribution



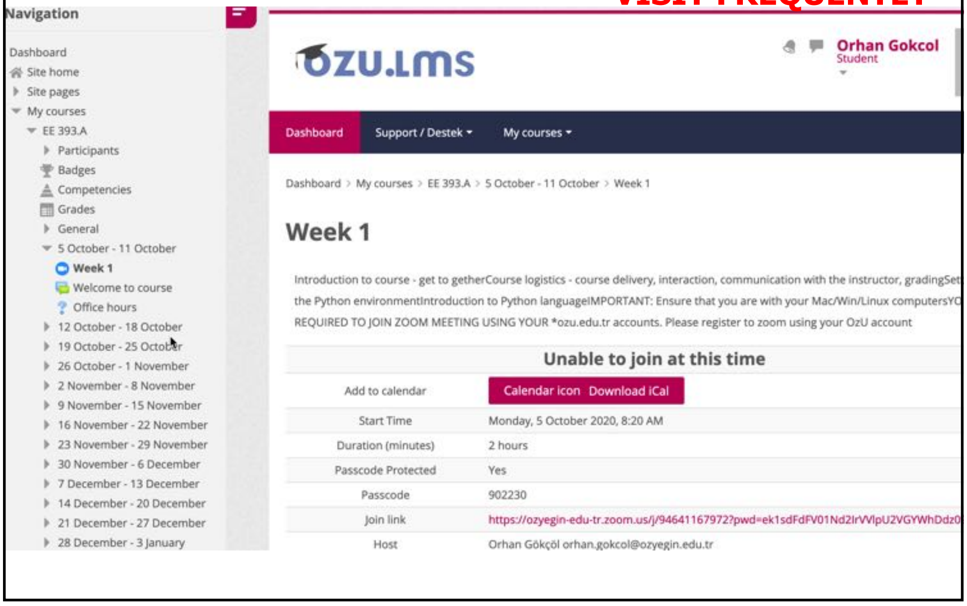
Points	100-90	89-85	84-80	79-75	74-70	69-66	65-62	61-58	57-54	53-50	0-49
Grade	A	A-	B+	B	B-	C+	C	C-	D+	D	F

Academic honesty is important. You will face with disciplinary actions for the cases against academic honesty

8

Course Resources

VISIT FREQUENTLY



Dashboard > My courses > EE 393.A > 5 October - 11 October > Week 1

Week 1

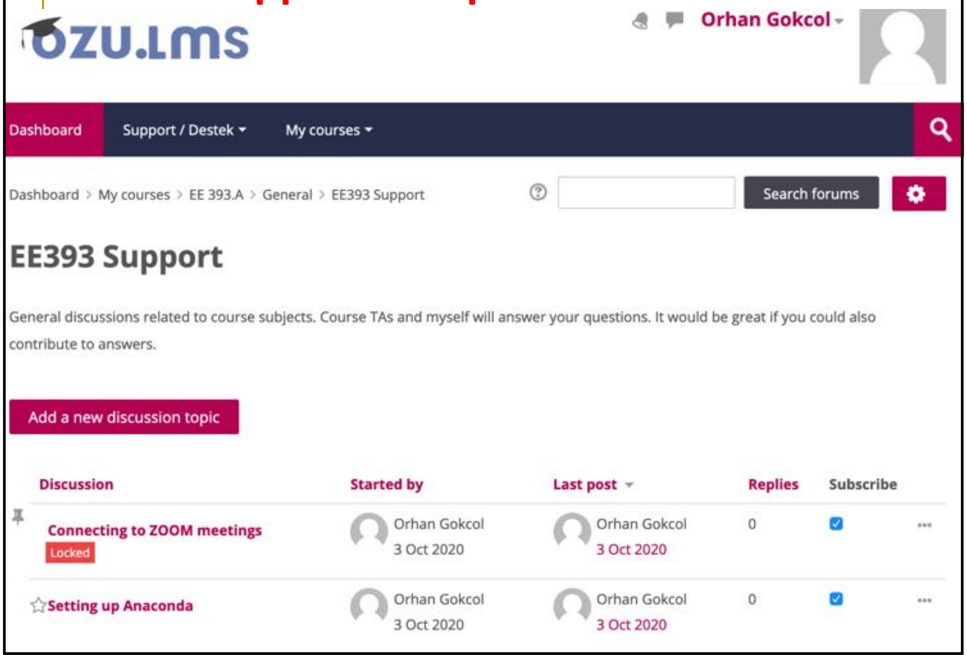
Introduction to course - get to getherCourse logistics - course delivery, interaction, communication with the instructor, gradingSet
the Python environmentIntroduction to Python languageIMPORTANT: Ensure that you are with your Mac/Win/Linux computersYO
REQUIRED TO JOIN ZOOM MEETING USING YOUR *ozu.edu.tr accounts. Please register to zoom using your OzU account

Unable to join at this time

Add to calendar	Calendar icon Download iCal
Start Time	Monday, 5 October 2020, 8:20 AM
Duration (minutes)	2 hours
Passcode Protected	Yes
Passcode	902230
Join link	https://ozyegin-edu-tr.zoom.us/j/94641167972?pwd=ek1sdFdFV01Nd2lrVVlpU2VGYWhDdz0
Host	Orhan Gökçöl orhan.gokcol@ozyegin.edu.tr

9

Course support –Helpdesk Forum



Dashboard > My courses > EE 393.A > General > EE393 Support

EE393 Support

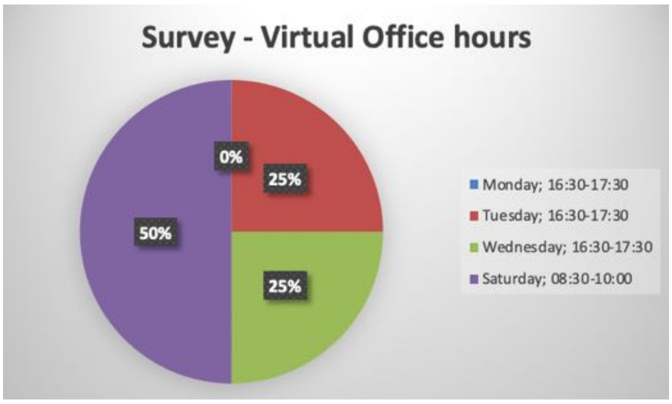
General discussions related to course subjects. Course TAs and myself will answer your questions. It would be great if you could also contribute to answers.

Add a new discussion topic

Discussion	Started by	Last post	Replies	Subscribe	
Connecting to ZOOM meetings Locked	Orhan Gokcol 3 Oct 2020	Orhan Gokcol 3 Oct 2020	0	☑	...
☆Setting up Anaconda	Orhan Gokcol 3 Oct 2020	Orhan Gokcol 3 Oct 2020	0	☑	...

10

Course support – Virtual Office Hour

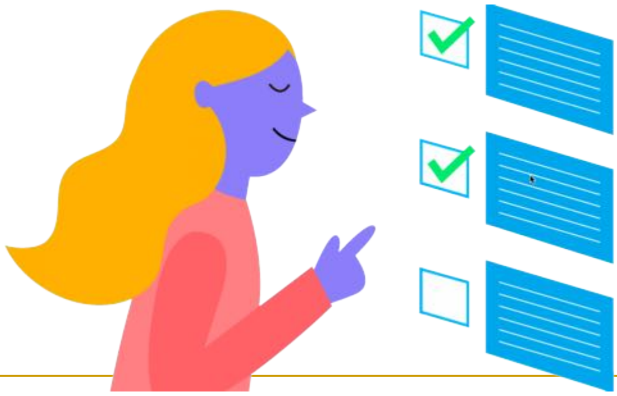


(ZOOM)

11

Survey….

- Please respond to survey. It contains five questions, and will not take more than a minute!!!



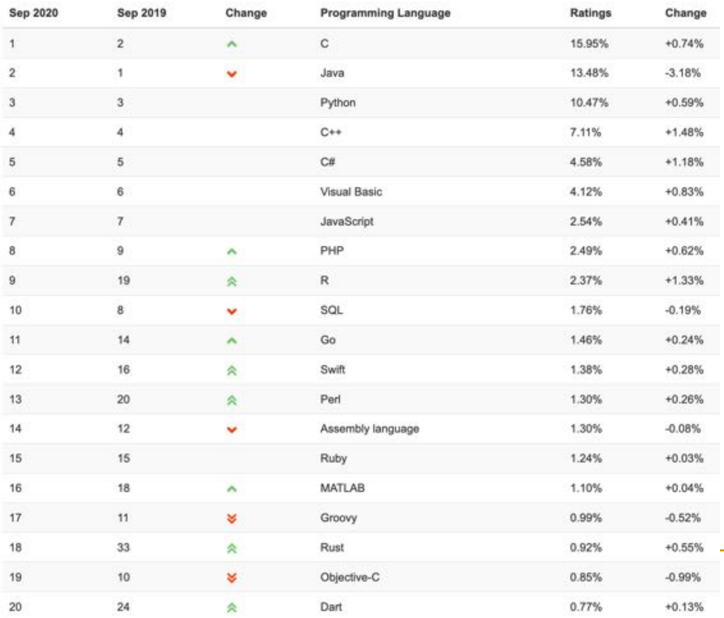
12

Rules for online classes

- Be ready before the class starts! (i.e. before 08:40)
- Microphones are muted by default. Do not unmute unless you are permitted.
- **It is not permitted to**
  - record the class video or audio
  - stream the class through social media or web
  - take pictures of the screens
- If you have a question, please type your question in the chat area. While I answer your question, you may unmute your mic & talk to me.
- You may prefer not to turn your camera on. It is OK for me. However, I prefer you turn it on.
- Respect the privacy of your friends and me.
- OzU Distance learning regulations are applied!!

13

Python is a popular language



Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%
11	14	↑	Go	1.46%	+0.24%
12	16	⇈	Swift	1.38%	+0.28%
13	20	⇈	Perl	1.30%	+0.26%
14	12	↓	Assembly language	1.30%	-0.08%
15	15		Ruby	1.24%	+0.03%
16	18	↑	MATLAB	1.10%	+0.04%
17	11	⇊	Groovy	0.99%	-0.52%
18	33	⇈	Rust	0.92%	+0.55%
19	10	⇊	Objective-C	0.85%	-0.99%
20	24	⇈	Dart	0.77%	+0.13%

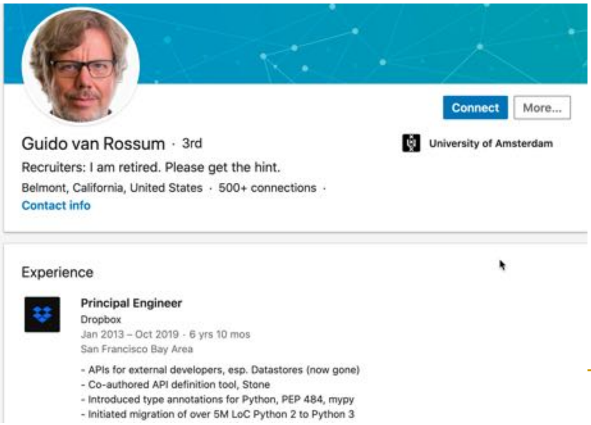
Ref:
www.tiobe.com

14

Python….

Python was created in the early 1990s by **Guido van Rossum** at Stichting Mathematisch Centrum in the NL as a successor of a language called ABC. Guido remains Python’s principal author, although it includes many contributions from others.

He **named** it **after** the television show Monty **Python's** Flying Circus.



Guido van Rossum · 3rd

University of Amsterdam

Recruiters: I am retired. Please get the hint.

Belmont, California, United States · 500+ connections ·

Contact info

Experience

Principal Engineer
Dropbox
Jan 2013 – Oct 2019 · 6 yrs 10 mos
San Francisco Bay Area

- APIs for external developers, esp. Datastores (now gone)
- Co-authored API definition tool, Stone
- Introduced type annotations for Python, PEP 484, mypy
- Initiated migration of over 5M LoC Python 2 to Python 3

(As of Oct. 2020)

Where do we use Python?

- Web and internet programming & development
  - Frameworks: Djongo, flask
  - Standard library supporting TCP/IP protocols (http, smtp etc.)
- Scientific and numeric computing
  - Libraries: SciPy, NumPy, MatPlotLib
  - Data analysis and modeling Lib: Pandas
- Machine learning
- Image Processing
- Digital Signal Processing
- GUI development for desktop and mobile
- Software development (build control, software testing)
- Scripting and shell programming
- Education (for teaching programming languages)

Everybody and every industry uses Python

Raspberry Pi

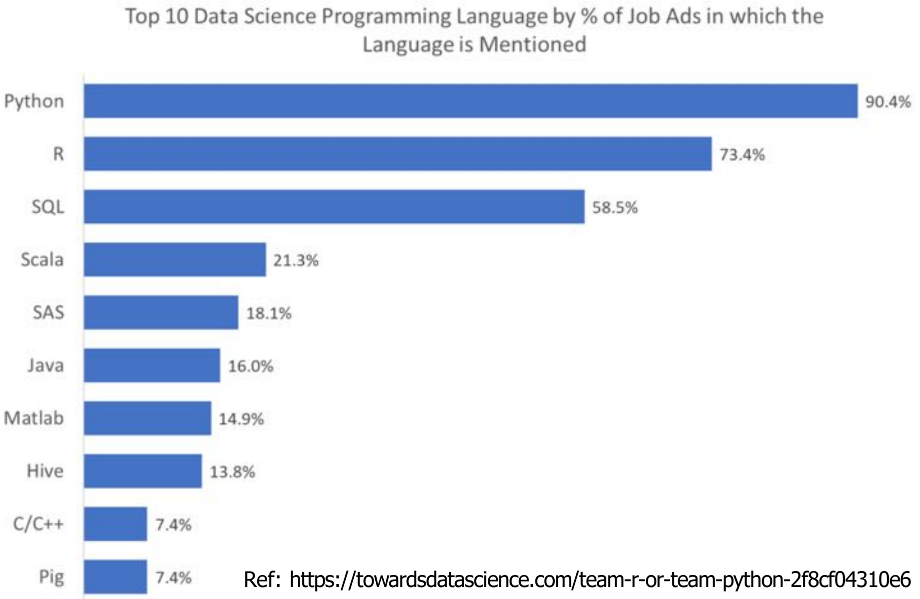
Instagram edX

reddit intel

Spotify Dropbox Microsoft Dropbox NASA YAHOO! IBM

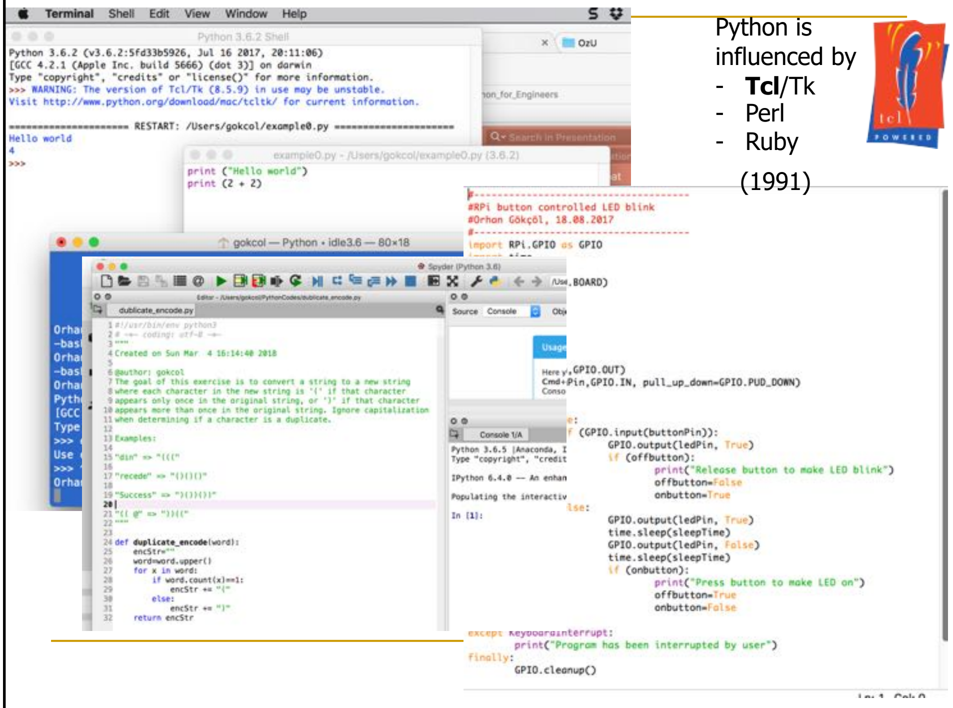
JPMorgan Chase & Co. NETFLIX amazon web services facebook YouTube MOZILLA Google

Python is popular in data science



Ref: https://towardsdatascience.com/team-r-or-team-python-2f8cf04310e6

17



```
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

===================== RESTART: /Users/gokcol/example0.py =====================
Hello world
4
>>>
```

```
print ("Hello world")
print (2 + 2)
```

```
Orha
-bas
Orha
-bas
Orha
Pyth
[GCC
Type
>>>
Use
>>>
Orha
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Mar  4 16:14:40 2018

@author: gokcol
The goal of this exercise is to convert a string to a new string
where each character in the new string is '(' if that character
appears only once in the original string, or ')' if that character
appears more than once in the original string. Ignore capitalization
when determining if a character is a duplicate.

Examples:

"din" => "((("

"recede" => "()()()"

"Success" => ")())())"

"(( @" => "))(("
"""

def duplicate_encode(word):
    encStr=""
    word=word.upper()
    for x in word:
        if word.count(x)==1:
            encStr += "("
        else:
            encStr += ")"
    return encStr
```

```
Python 3.6.5 |Anaconda, I
Type "copyright", "credit

IPython 6.4.0 -- An enhan

Populating the interactiv

In [1]:
```

```
#-----------------------------------------
#RPi button controlled LED blink
#Orhan Gökçöl, 18.08.2017
#-----------------------------------------
import RPi.GPIO as GPIO
                     .BOARD)

                     ,GPIO.OUT)
                     Pin,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

        e:
        f (GPIO.input(buttonPin)):
              GPIO.output(ledPin, True)
              if (offbutton):
                     print("Release button to make LED blink")
                     offbutton=False
                     onbutton=True
        lse:
              GPIO.output(ledPin, True)
              time.sleep(sleepTime)
              GPIO.output(ledPin, False)
              time.sleep(sleepTime)
              if (onbutton):
                     print("Press button to make LED on")
                     offbutton=True
                     onbutton=False
except KeyboardInterrupt:
       print("Program has been interrupted by user")
finally:
       GPIO.cleanup()
```

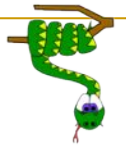
Python is influenced by

- **Tcl**/Tk
- Perl
- Ruby

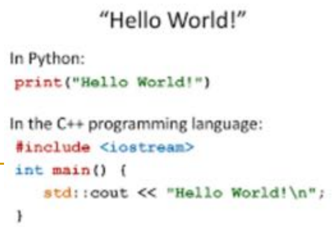
(1991)

18

Why Python?



- Productivity –fast coding, very fast!
- Open source and community driven
- Large collection of support libraries
- Coherence
  - Ease of use and learn –almost as easy as speaking English (!)
  - Rapid learning curve
  - Easy code maintenance
  - Type-safe – no need to declare variable types
  - Interpreted; but compilation is possible
  - Rapid development
- IoT applications
- Big Data
- Machine Learning



"Hello World!"

In Python:

```
print("Hello World!")
```

In the C++ programming language:

```
#include <iostream>
int main() {
    std::cout << "Hello World!\n";
}
```

19

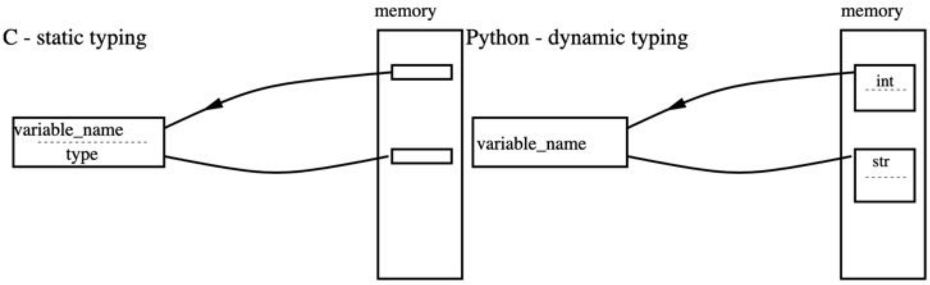
Important features

no compiling or linking	rapid development cycle
no type declarations	simpler, shorter, more flexible
automatic memory management	garbage collection
high-level data types and operations	fast development
object-oriented programming	code structuring and reuse, C++
embedding and extending in C	mixed language systems
classes, modules, exceptions	"programming-in-the-large" support
dynamic loading of C modules	simplified extensions, smaller binaries
dynamic reloading of C modules	programs can be modified without stopping

20

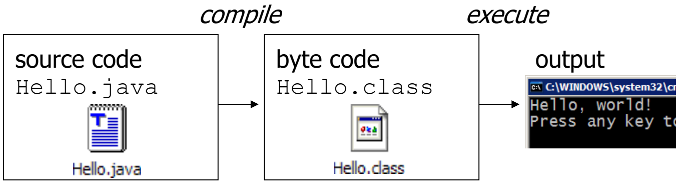
Python is an interpreted language with dynamic typing

- This means that, Python programs are executed line by line, from top to bottom, and we don’t declare types for the variables



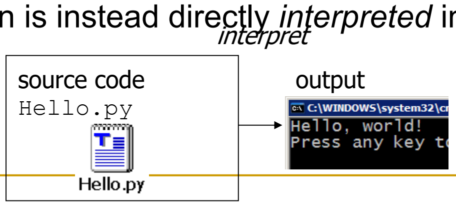
Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



```
Hello, world!
Press any key t
```

- Python is instead directly *interpreted* into machine instructions.

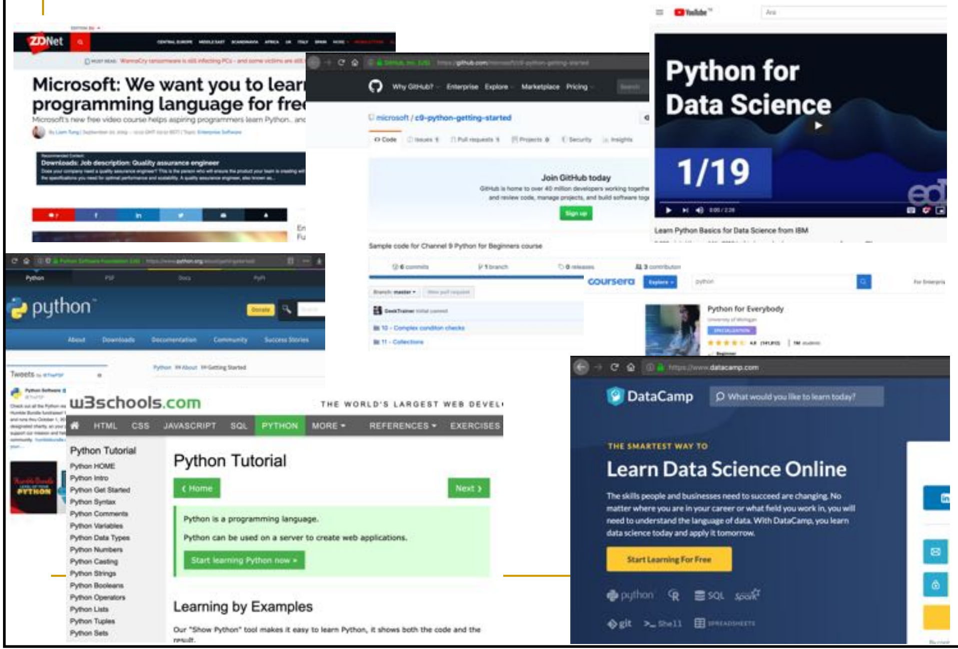


```
Hello, world!
Press any key t
```

It is possible to compile python

PyPy, JIT Compiler

Lots of online and free learning resources – See LMS


Microsoft: We want you to lear
programming language for fre

microsoft / c9-python-getting-started

Join GitHub today

Sample code for Channel 9 Python for Beginners course

Python for Data Science

1/19

Learn Python Basics for Data Science from IBM

Python for Everybody

w3schools.com

THE WORLD'S LARGEST WEB DEVEL

Python Tutorial

Python is a programming language.

Python can be used on a server to create web applications.

Start learning Python now »

Learning by Examples

Our "Show Python" tool makes it easy to learn Python, it shows both the code and the result.

DataCamp

What would you like to learn today?

THE SMARTEST WAY TO

Learn Data Science Online

The skills people and businesses need to succeed are changing. No matter where you are in your career or what field you work in, you will need to understand the language of data. With DataCamp, you learn data science today and apply it tomorrow.

Start Learning For Free

First things first:

- I won’t give you anything that you would be unable to find on the internet.
  - There are countless number of resources on the internet and you can learn Python all by yourself.
- Please see me as a "curator" who brings different resources together, manage and merge them with my experience in a theme.
- EE393 reflects my way about how I see the Python as a programming language for engineers. I will try to guide you through your learning experience. And, I will value your experience.

Official Warning

- This course requires your attention!!!!
- It is highly practical and programming intensive.
- I assumed you would require three to four hours of time per week as a minimum for EE393.

25

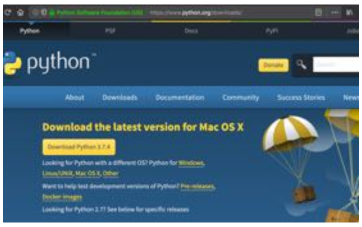
Python core language and extensions (libraries)

- Python core language consists of programming constructions and data structures.
- A **Python library** is a reusable chunk of code that you may want to include in your programs/ projects.
- Important libraries :
  - NumPy: Numerical Python, mathematics
  - SciPy: Scientific Python
  - Matplotlib: Graphics & Visualization
  - Pandas: Data analysis
  - Flask, Djongo: Web programming
  - Scikit-learn: Machine learning
  - Many other libraries in different fields

26

Installing Python :

- If you are on Mac or Linux, it is installed by default (most of the time!)



Download the latest version for Mac OS X

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

You also need to install a package manager for python : **pip** to have a hassle-free and up-to-date environment

- **Library updates**
- **Binary updates etc.**

www.python.org/downloads

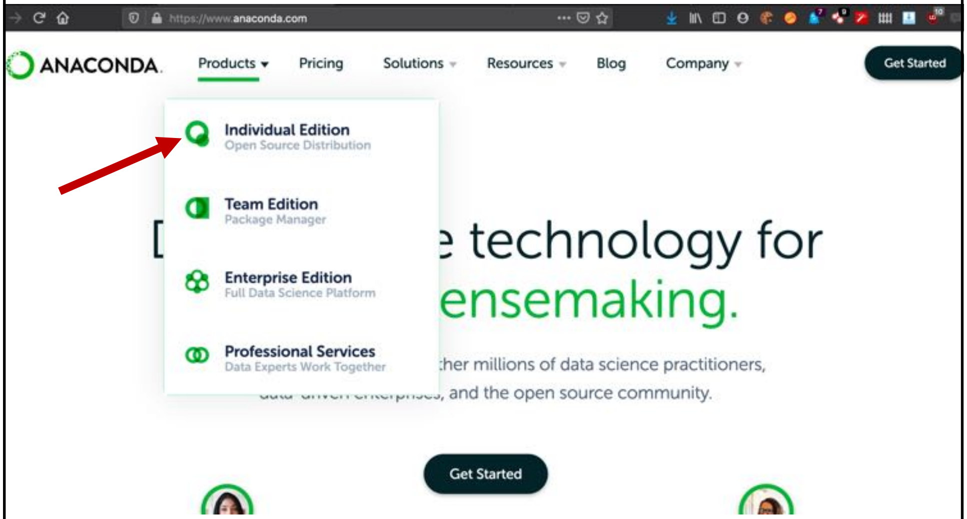
- Alternatively, you may use a python distribution packaged with engineering libraries. One of the most important distribution is Anaconda. This option is recommended.

ANACONDA

https://www.anaconda.com/

- Another option is to use Cloud environments. However, there are limitations like not able to accessing local file system and limitet resources availability.

Recommended Setup



[e technology for ensemaking.

her millions of data science practitioners, data-driven enterprises, and the open source community.

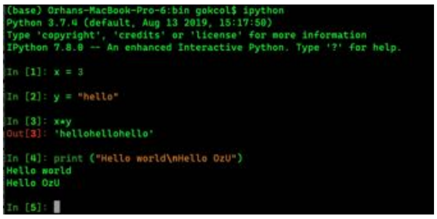
https://www.anaconda.com/

Python development environments

- Countless of possibilities: IDLE, Ipython, Shell, Text editors, PyCharm, Geddit, Eclipse, Xcode, Jupyter Notebooks, Anaconda, Visual Studio ….
  - However we mainly use two of them: **1) Jupyter Notebooks**, 2) Spyder
  - The trend in the software industry seems to go with Jupyter
- Several cloud computing environments (free for basic use! Be careful, you are sharing your data!)
  - Google colab (Jupyter)
  - Repl (Ipython)
  - Microsoft Azure Notebooks (practically Jupyter)
  - <…….>

IDLE

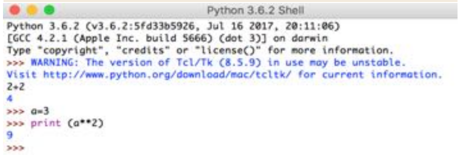
IDLE is the official Python "Shell" to interact with python and to write python programs. It is available in the standard python.



```
(base) Orhans-MacBook-Pro-6:bin gokcol$ ipython
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: x = 3

In [2]: y = "hello"

In [3]: x*y
Out[3]: 'hellohellohello'

In [4]: print ("Hello world\nHello OzU")
Hello world
Hello OzU

In [5]: 
```

Interactive Python (Ipython)

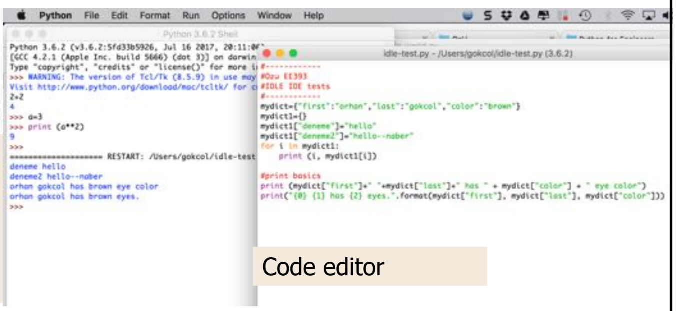
Interaction is possible (like Matlab)



```
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
2+2
4
>>> a=3
>>> print (a**2)
9
>>>
```

IDLE helps you program in Python by:

- color-coding your program code
- debugging
- auto-indent
- interactive shell



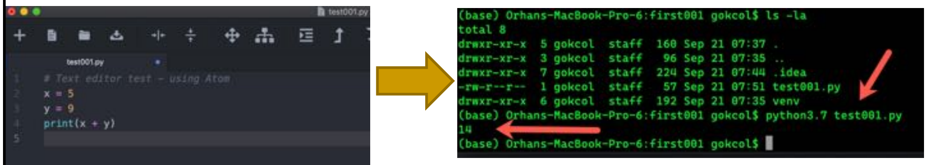
```
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more i
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may
Visit http://www.python.org/download/mac/tcltk/ for c
2+2
4
>>> a=3
>>> print (a**2)
9
>>>
==================== RESTART: /Users/gokcol/idle-test
deneme hello
deneme2 hello--naber
orhan gokcol has brown eye color
orhan gokcol has brown eyes.
>>>
```

```
#------------
#Ozu EE393
#IDLE IDE tests
#------------
mydict={"first":"orhan","last":"gokcol","color":"brown"}
mydict1={}
mydict1["deneme"]="hello"
mydict1["deneme2"]="hello--naber"
for i in mydict1:
    print (i, mydict1[i])

#print basics
print (mydict["first"]+" "+mydict["last"]+" has " + mydict["color"] + " eye color")
print("{0} {1} has {2} eyes.".format(mydict["first"], mydict["last"], mydict["color"]))
```

Code editor

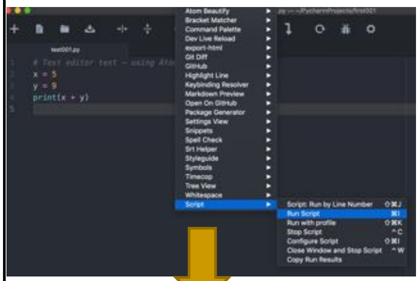
(We won't use IDLE; but it is the base development environment)

Using a text editor (such as Atom) to write Python code; then run from the command shell

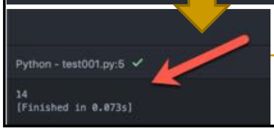


```
# Text editor test - using Atom
x = 5
y = 9
print(x + y)
```

```
(base) Orhans-MacBook-Pro-6:first001 gokcol$ ls -la
total 8
drwxr-xr-x  5 gokcol  staff  160 Sep 21 07:37 .
drwxr-xr-x  3 gokcol  staff   96 Sep 21 07:35 ..
drwxr-xr-x  7 gokcol  staff  224 Sep 21 07:44 .idea
-rw-r--r--  1 gokcol  staff   57 Sep 21 07:51 test001.py
drwxr-xr-x  6 gokcol  staff  192 Sep 21 07:35 venv
(base) Orhans-MacBook-Pro-6:first001 gokcol$ python3.7 test001.py
14
(base) Orhans-MacBook-Pro-6:first001 gokcol$ 
```



You can install several other packages to Atom to ease Python development such as "hydrogen" and "termination"



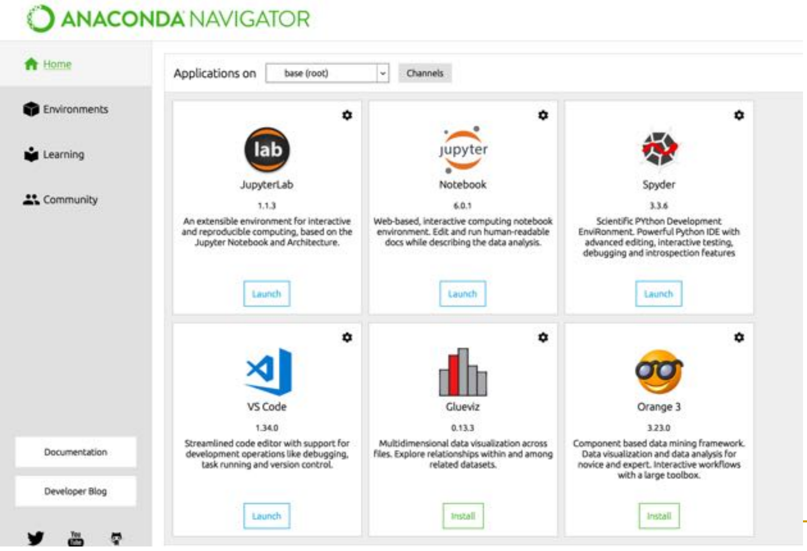
```
Python - test001.py:5 ✓

14
[Finished in 0.073s]
```

(You must install "script" package

31

Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment.



JupyterLab
1.1.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

VS Code
1.34.0
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch

Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

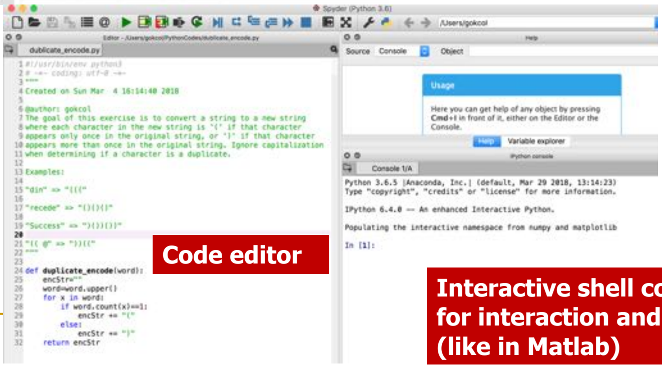
Orange 3
3.23.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

A working environment with a package manager : **conda**

32

Anaconda Navigator & spyder

Anaconda is a python distribution, with **installation** and **package management** tools. It provides large selection of packages and commercial support. It is an environment manager, which provides the facility to create different python environments, each with their own settings.



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Mar  4 16:14:40 2018

@author: gokcol
The goal of this exercise is to convert a string to a new string
where each character in the new string is '(' if that character
appears only once in the original string, or ')' if that character
appears more than once in the original string. Ignore capitalization
when determining if a character is a duplicate.

Examples:

"din" => "((("

"recede" => "()()()"

"Success" => ")())())"

"(( @" => "))(("
"""

def duplicate_encode(word):
    encStr=""
    word=word.upper()
    for x in word:
        if word.count(x)==1:
            encStr += "("
        else:
            encStr += ")"
    return encStr
```

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

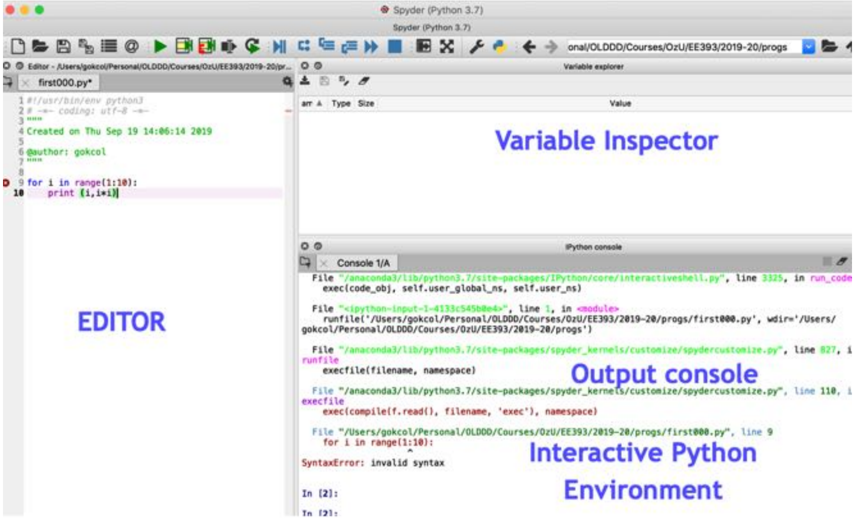
```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:14:23)
Type "copyright", "credits" or "license" for more information.

IPython 6.4.0 -- An enhanced Interactive Python.

Populating the interactive namespace from numpy and matplotlib

In [1]:
```

Code editor

Interactive shell console for interaction and output (like in Matlab)

Spyder

I will use Spyder



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 19 14:06:14 2019

@author: gokcol
"""

for i in range(1:10):
    print (i,i*i)
```

```
  File "/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py", line 3325, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-1-4133c545b0e4>", line 1, in <module>
    runfile('/Users/gokcol/Personal/OLDDD/Courses/OzU/EE393/2019-20/progs/first000.py', wdir='/Users/
gokcol/Personal/OLDDD/Courses/OzU/EE393/2019-20/progs')

  File "/anaconda3/lib/python3.7/site-packages/spyder_kernels/customize/spydercustomize.py", line 827, i
runfile
    execfile(filename, namespace)

  File "/anaconda3/lib/python3.7/site-packages/spyder_kernels/customize/spydercustomize.py", line 110, i
execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "/Users/gokcol/Personal/OLDDD/Courses/OzU/EE393/2019-20/progs/first000.py", line 9
    for i in range(1:10):
                    ^
SyntaxError: invalid syntax


In [2]:

In [2]:
```

Spyder (Python 3.7)
Spyder (Python 3.7)
onal/OLDDD/Courses/OzU/EE393/2019-20/progs
Editor - /Users/gokcol/Personal/OLDDD/Courses/OzU/EE393/2019-20/pr...
first000.py*
Variable explorer
Type
Size
Value
IPython console
Console 1/A

EDITOR

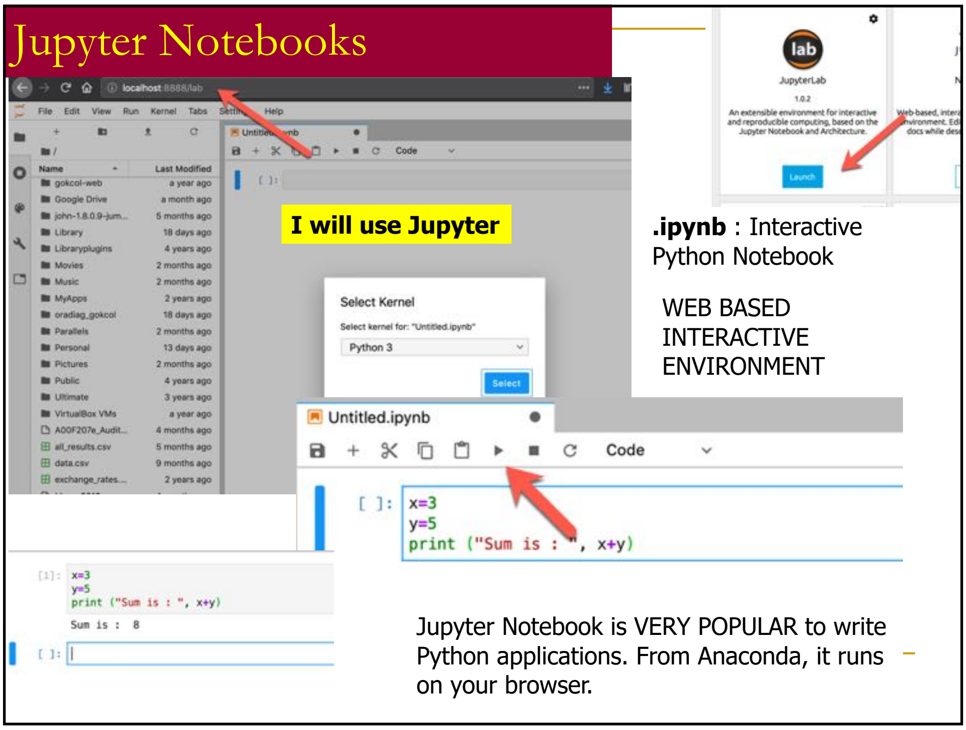
Variable Inspector

Output console

Interactive Python Environment

(It is also available as a stand alone editor)

Jupyter Notebooks



JupyterLab

1.0.2

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Name	Last Modified
gokcol-web	a year ago
Google Drive	a month ago
john-1.8.0.9-jum...	5 months ago
Library	18 days ago
Libraryplugins	4 years ago
Movies	2 months ago
Music	2 months ago
MyApps	2 years ago
oradiag_gokcol	18 days ago
Parallels	2 months ago
Personal	13 days ago
Pictures	2 months ago
Public	4 years ago
Ultimate	3 years ago
VirtualBox VMs	a year ago
A00F207e_Audit...	4 months ago
all_results.csv	5 months ago
data.csv	9 months ago
exchange_rates....	2 years ago

I will use Jupyter

Select Kernel

Select kernel for: "Untitled.ipynb"

Python 3

```
[ ]: x=3
     y=5
     print ("Sum is : ", x+y)
```

```
[1]: x=3
     y=5
     print ("Sum is : ", x+y)

     Sum is :  8

[ ]:
```

.ipynb : Interactive Python Notebook

WEB BASED
INTERACTIVE
ENVIRONMENT

Jupyter Notebook is VERY POPULAR to write Python applications. From Anaconda, it runs on your browser.

35

Jupyter exercises

- <Follow the instructions of your instructor>

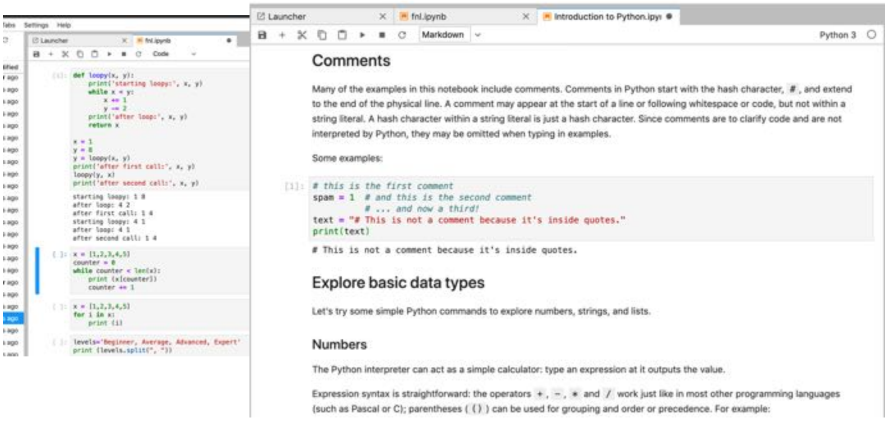
TASKS

- Get familiarized with Jupyter
- Use Python ac a calculator
- Experiment on variables and print statement

36

Jupyter examples

Download from OzU LMS /week 1



```
[1]: def loopy(x, y):
         print('starting loopy:', x, y)
         while x < y:
             x += 1
             y -= 2
         print('after loop:', x, y)
         return x

     x = 1
     y = 8
     y = loopy(x, y)
     print('after first call:', x, y)
     loopy(y, x)
     print('after second call:', x, y)
```

```
starting loopy: 1 8
after loop: 4 2
after first call: 1 4
starting loopy: 4 1
after loop: 4 1
after second call: 1 4
```

```
[ ]: x = [1,2,3,4,5]
     counter = 0
     while counter < len(x):
         print (x[counter])
         counter += 1
```

```
[ ]: x = [1,2,3,4,5]
     for i in x:
         print (i)
```

```
[ ]: levels='Beginner, Average, Advanced, Expert'
     print (levels.split(", "))
```

Comments

Many of the examples in this notebook include comments. Comments in Python start with the hash character, #, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

Some examples:

```
[1]: # this is the first comment
     spam = 1  # and this is the second comment
               # ... and now a third!
     text = "# This is not a comment because it's inside quotes."
     print(text)
```

```
# This is not a comment because it's inside quotes.
```

Explore basic data types

Let's try some simple Python commands to explore numbers, strings, and lists.

Numbers

The Python interpreter can act as a simple calculator: type an expression at it outputs the value.

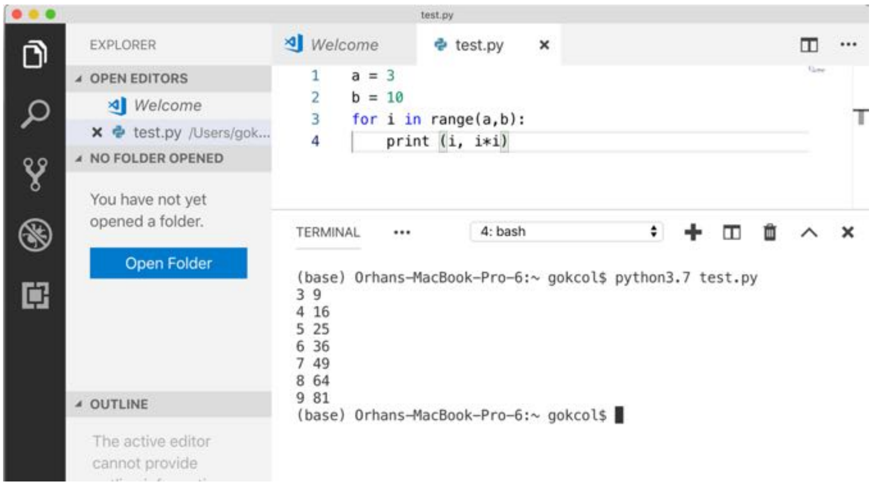
Expression syntax is straightforward: the operators +, -, * and / work just like in most other programming languages (such as Pascal or C); parentheses (()) can be used for grouping and order or precedence. For example:

We will go through an example Jupyter notebook which covers Fundamentals of Python (courtesy of Microsoft)

Ref: https://notebooks.azure.com/Microsoft/projects/2018-Intro-Python/html/Introduction%20to%20Python.ipynb

37

Visual Studio Code

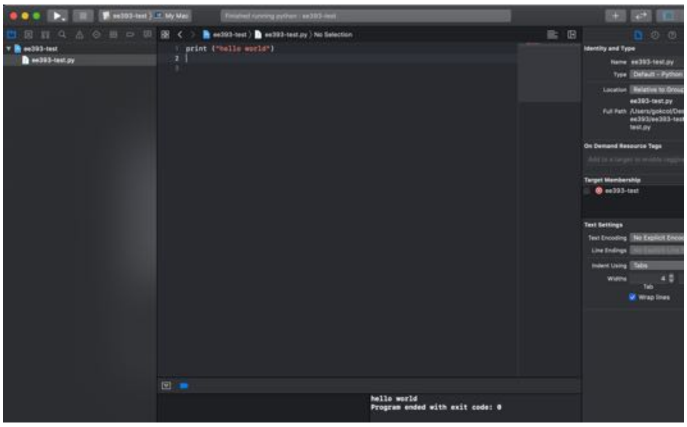


```
a = 3
b = 10
for i in range(a,b):
    print (i, i*i)
```

```
(base) Orhans-MacBook-Pro-6:~ gokcol$ python3.7 test.py
3 9
4 16
5 25
6 36
7 49
8 64
9 81
(base) Orhans-MacBook-Pro-6:~ gokcol$ ▌
```

38

Mac OS XCode

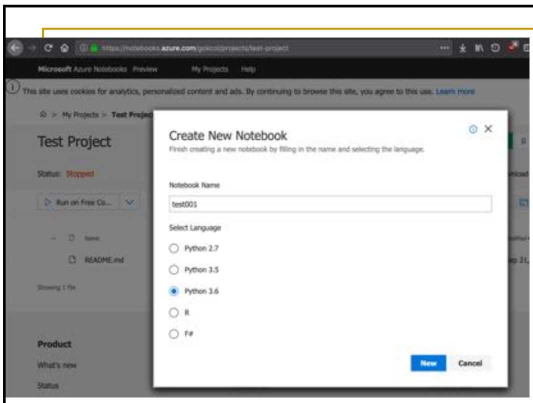


```
print ("hello world")
```

```
hello world
Program ended with exit code: 0
```

You need some manual adjustments. See
https://www.youtube.com/watch?v=GUBGoeCu19I
for instructions

Azure Notebooks



Create New Notebook

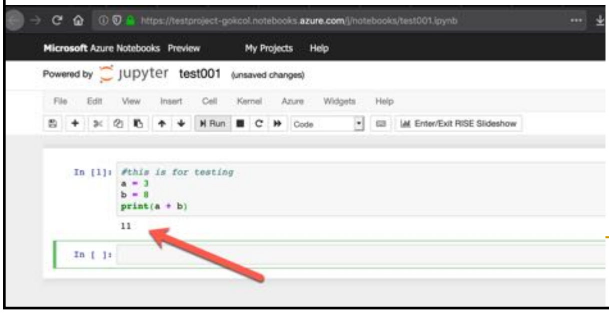
Finish creating a new notebook by filling in the name and selecting the language.

Notebook Name: test001

Select Language

○ Python 2.7
○ Python 3.5
● Python 3.6
○ R
○ F#

It is a cloud computing environment. You can log in to **notebooks.azure.com** Using your student email account



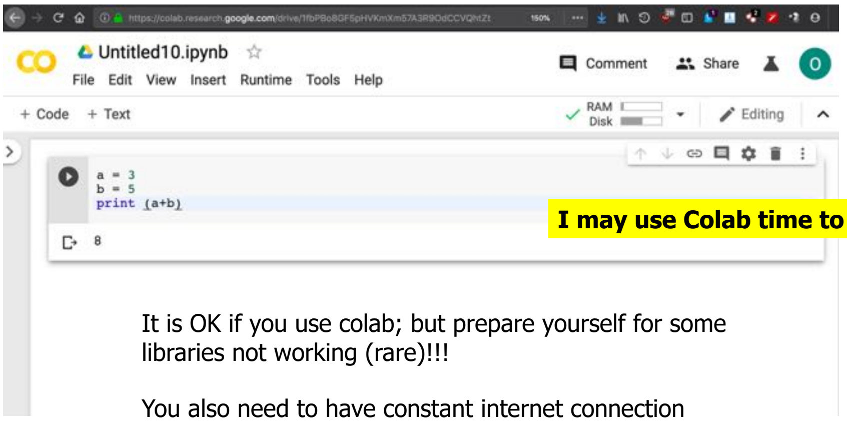
https://testproject-gokcol.notebooks.azure.com/j/notebooks/test001.ipynb

Microsoft Azure Notebooks Preview
My Projects
Help
Powered by jupyter test001 (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Azure
Widgets
Help
Run
Code
Enter/Exit RISE Slideshow

```
In [1]: #this is for testing
        a = 3
        b = 8
        print(a + b)

        11

In [ ]:
```

Sometimes web site crashes or responds slowly!!

Google colab: http://colab.research.google.com


https://colab.research.google.com/drive/1fbPBoBGF5pHVKmXm57A3R9OdCCVQhtZt
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
a = 3
b = 5
print (a+b)
```

```
8
```

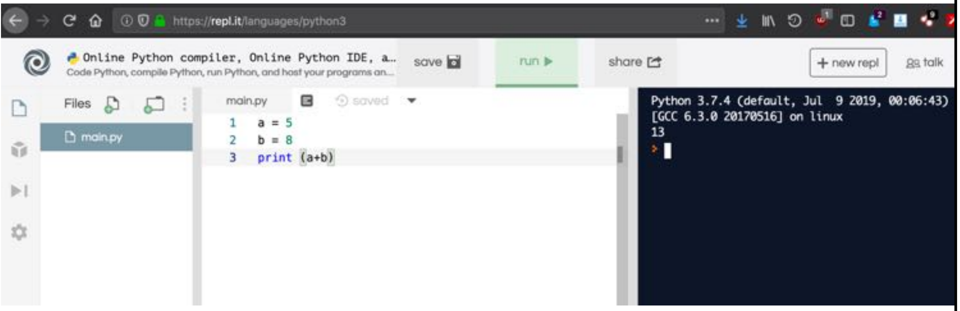
I may use Colab time to time

It is OK if you use colab; but prepare yourself for some libraries not working (rare)!!!

You also need to have constant internet connection

Sometimes web site crashes or responds slowly!!

Repl


https://repl.it/languages/python3
Online Python compiler, Online Python IDE, a...
Code Python, compile Python, run Python, and host your programs an...
save
run ▶
share
+ new repl
talk
Files
main.py
main.py
saved

```
a = 5
b = 8
print (a+b)
```

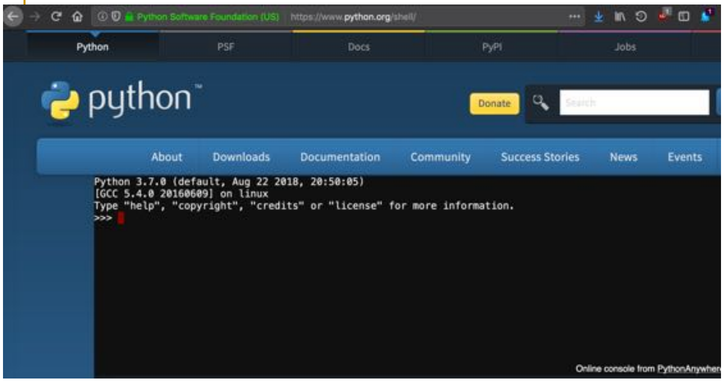
```
Python 3.7.4 (default, Jul  9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
13
> █
```

It is OK if you use Repl; but prepare yourself for some libraries not working!!!

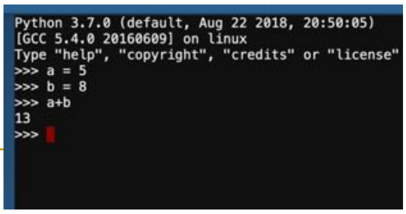
You also need to have constant internet connection

Sometimes web site crashes (more than colab!!)

Python anywhere: www.python.org/shell



```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```



```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license"
>>> a = 5
>>> b = 8
>>> a+b
13
>>> 
```

(Don't use; but it is ok for learning)

Some tasks till next week:

- Install Python 3.8.x
- Experiment with IDLE interface and write simple Python programs and run it within IDLE
- Install anaconda (www.anaconda.com) and experiment with Spyder and Jupyter.
- Simplest rules for Python codes:
  - No semicolon is needed at the end of each statement
  - No type declaration (i.e. no int, no float etc). Just use the variables
  - Same arithmetic operators (+,-,*,/,**(=power))
  - Use print() method to print something on to console
  - Strings are anything you give in "….."
  - Experiment some string methods

```
myString="hello world"
myString.title()
myString.upper()
myString.isdigit()
myString.islower()
```

Basic Rules for operations between variables

- **expression**: A data value or set of operations to compute a value.

  Examples: `1 + 4 * 3`

  `42`

All operators work with integer and real numbers. Expression result is integer if all operands are integers, and is real if one of the operands is real number.

- Arithmetic operators we will use:

`+ - * /`	addition, subtraction/negation, multiplication, division
`%`	modulus, (i.e. remainder)
`**`	exponentiation
//	integer division (discard remainder)

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

    `1 + 3 * 4` is `13`
  - **Parentheses** can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

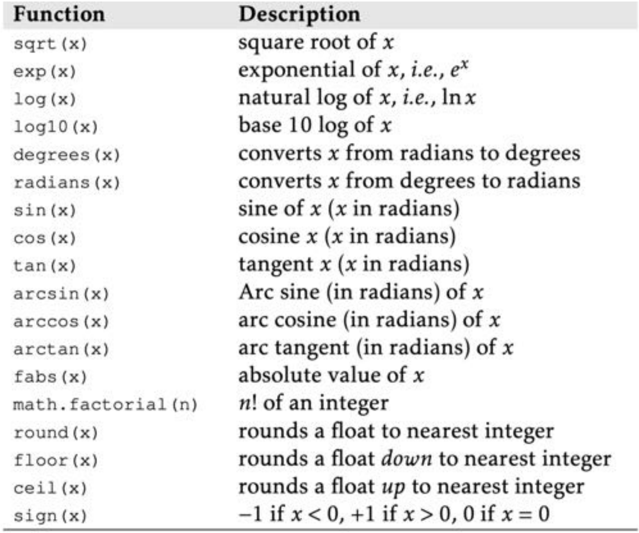
See
https://stackoverflow.com/questions/15193927/what-do-these-operators-mean

45

Math commands

- Some built-in commands to perform scientific calculations:
  - To use many of these commands, you must write the following at the top of your Python program:

```
from math import *
```



Function	Description
`sqrt(x)`	square root of x
`exp(x)`	exponential of x, *i.e.*, e^x
`log(x)`	natural log of x, *i.e.*, $\ln x$
`log10(x)`	base 10 log of x
`degrees(x)`	converts x from radians to degrees
`radians(x)`	converts x from degrees to radians
`sin(x)`	sine of x (x in radians)
`cos(x)`	cosine x (x in radians)
`tan(x)`	tangent x (x in radians)
`arcsin(x)`	Arc sine (in radians) of x
`arccos(x)`	arc cosine (in radians) of x
`arctan(x)`	arc tangent (in radians) of x
`fabs(x)`	absolute value of x
`math.factorial(n)`	$n!$ of an integer
`round(x)`	rounds a float to nearest integer
`floor(x)`	rounds a float *down* to nearest integer
`ceil(x)`	rounds a float *up* to nearest integer
`sign(x)`	-1 if $x < 0$, $+1$ if $x > 0$, 0 if $x = 0$

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

46

Variables

- **variable**: A named piece of memory that can store a value.
  - Usage:
    - Compute an expression's result,
    - store that result into a variable,
    - and use that variable later in the program.
- **assignment statement**: Stores a value into a variable.
  - Syntax:

    name = ***value***

    Usual variable naming rules apply!!! (same as C, Java etc.)
  - Examples:

    ```
    x = 5
    gpa = 3.14
    ```
  - A variable that has been given a value can be used in expressions.

    `x + 4` is `9`

47

Reserved Words

- You can not use reserved words as variable names / identifiers

and del for is raise
assert elif from lambda return
break else global not try
class except if or while
continue exec import pass yield
def finally in print

All reserved words are in small letter!!!!!

48

Use variable names "wisely" !!!!!!!!!

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print (x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print (c)
```

What is this code doing?

```
hours = 35.0
rate = 12.50
pay = hours * rate
print pay
```

49

print

- `print` : Produces text output on the console.
- Syntax:

  `print ("`***Message***`”)`

  `print (`***Expression)***

  - Prints the given text message or expression value on the console, and moves the cursor down to the next line.

  `print (`***Item1***`,` ***Item2***`,` ...`,` ***ItemN)***

  - Prints several messages and/or expressions on the same line.

- Examples:

```
print ("Hello, world!”)
age = 45
print ("You have", 65 - age, "years until retirement”)
```

Output:

```
Hello, world!
You have 20 years until retirement
```

50

input

- `input` : Reads a number from user input.
  - You can assign (store) the result of `input` into a variable.
  - Example:

```
age = input("How old are you? ")
print ("Your age is", age)
print ("You have", 65 - int(age), "years
until retirement”)
```

Output:

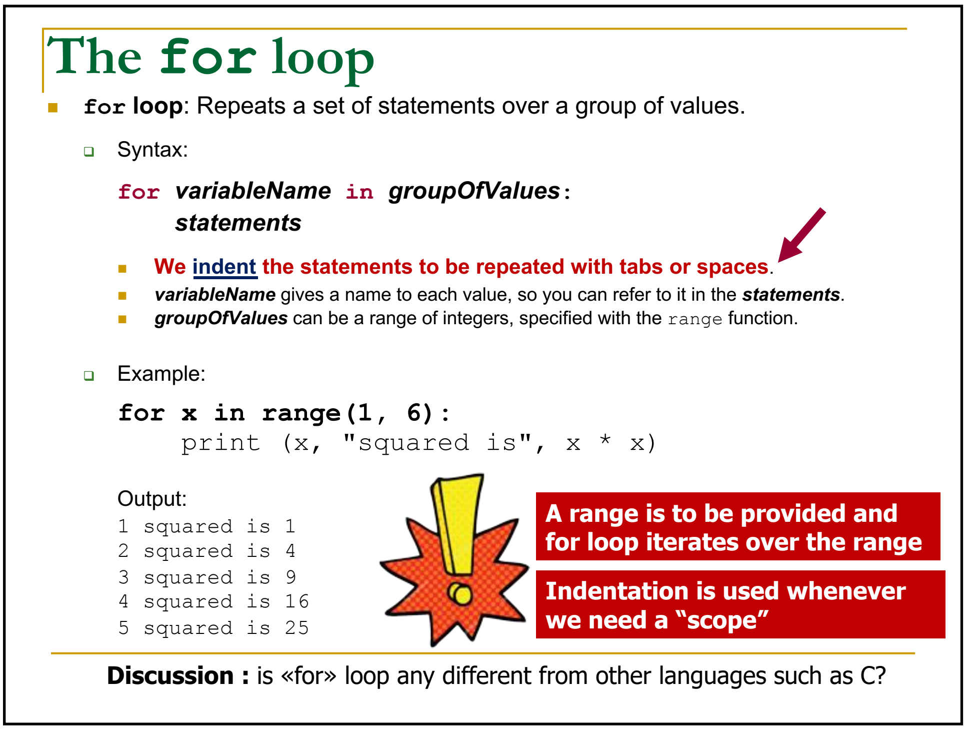
```
How old are you? 53
Your age is 53
You have 12 years until retirement
```

51

Comments in Python

- Anything after a **#** is ignored by Python
- Why comment?
  - Describe what is going to happen in a sequence of code
  - Document who wrote the code or other ancillary information
  - Turn off a line of code - perhaps temporarily

52



The `for` loop

- `for` **loop**: Repeats a set of statements over a group of values.
  - Syntax:

```
for variableName in groupOfValues:
    statements
```

    - **We indent the statements to be repeated with tabs or spaces**.
    - ***variableName*** gives a name to each value, so you can refer to it in the ***statements***.
    - ***groupOfValues*** can be a range of integers, specified with the `range` function.
  - Example:

```
for x in range(1, 6):
    print (x, "squared is", x * x)
```

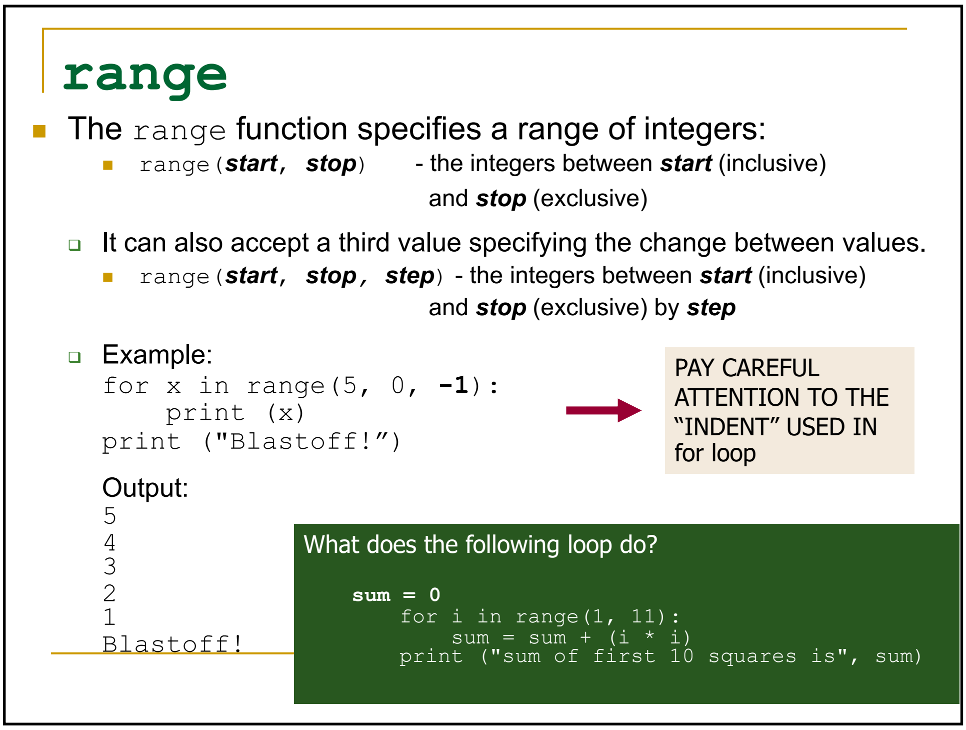
Output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
```

A range is to be provided and for loop iterates over the range

Indentation is used whenever we need a "scope"

Discussion : is «for» loop any different from other languages such as C?



`range`

- The `range` function specifies a range of integers:
  - `range(`***start***`,` ***stop***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive)
- It can also accept a third value specifying the change between values.
  - `range(`***start***`,` ***stop***`,` ***step***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive) by ***step***
- Example:

```
for x in range(5, 0, -1):
    print (x)
print ("Blastoff!")
```

PAY CAREFUL ATTENTION TO THE "INDENT" USED IN for loop

Output:

```
5
4
3
2
1
Blastoff!
```

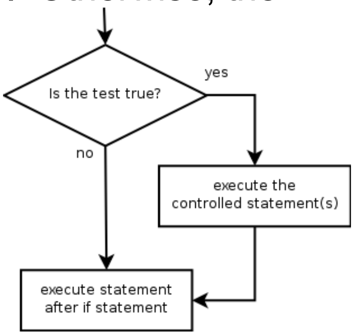
What does the following loop do?

```
sum = 0
    for i in range(1, 11):
        sum = sum + (i * i)
    print ("sum of first 10 squares is", sum)
```

if

- **`if` statement**: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.
  - Syntax:

```
if condition:
        statements
```



Is the test true?
yes
no
execute the controlled statement(s)
execute statement after if statement

- Example:

PAY CAREFUL ATTENTION TO THE "INDENT" USED in if block

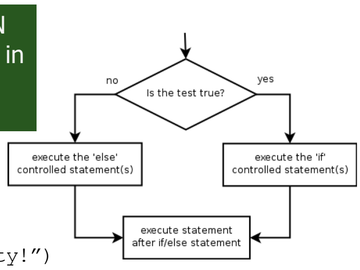
```
gpa = 3.4
if gpa > 2.0:
    print "Your application is accepted."
```

if/else

- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if condition:
        statements
else:
        statements
```

PAY CAREFUL ATTENTION TO THE "INDENT"s USED in the scopes in if/else/elif block



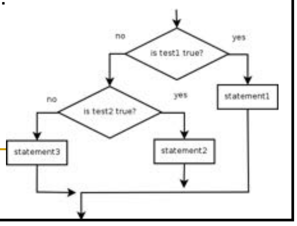
Is the test true?
no
yes
execute the 'else' controlled statement(s)
execute the 'if' controlled statement(s)
execute statement after if/else statement

- Example:

```
gpa = 1.4
if gpa > 2.0:
    print ("Welcome to Ozyegin University!”)
else:
    print ("Your application is denied.”)
```

- Multiple conditions can be chained with `elif` ("else if"):

```
if condition:
        statements
elif condition:
        statements
else:
        statements
```



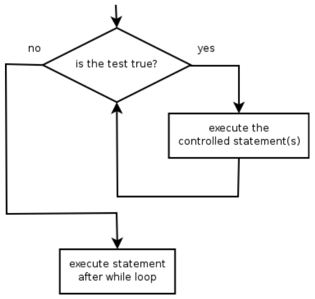
while

- Executes a group of statements as long as a condition is True.
  - good for *indefinite loops* (repeat an unknown number of times)
- Syntax:

```
while condition:
    statements
```

- Example:

```
number = 1
while number < 200:
    print (number, end=" ")
    number = number * 2
```


no
yes
is the test true?
execute the controlled statement(s)
execute statement after while loop

  - Output:

```
1 2 4 8 16 32 64 128
```

57

Logic

- Many logical expressions use *relational operators*:

Operator	Meaning	Example	Result
`==`	equals	`1 + 1 == 2`	`True`
`!=`	does not equal	`3.2 != 2.5`	`True`
`<`	less than	`10 < 5`	`False`
`>`	greater than	`10 > 5`	`True`
`<=`	less than or equal to	`126 <= 100`	`False`
`>=`	greater than or equal to	`5.0 >= 5.0`	`True`

- Logical expressions can be combined with *logical operators*:

Operator	Example	Result
`and`	`9 != 6 and 2 < 3`	`True`
`or`	`2 == 3 or -1 < 5`	`True`
`not`	`not 7 > 0`	`False`

58

Strings

- **string**: A sequence of text characters in a program.
  - Strings start and end with quotation mark " or apostrophe ' characters.
  - Examples:

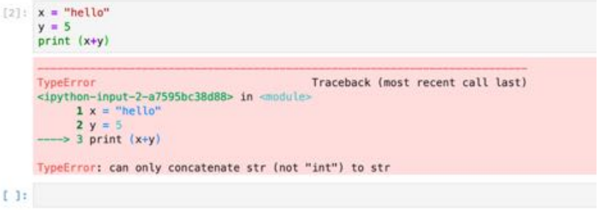
```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

- A string may not span across multiple lines or contain a " character.

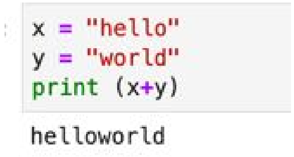
```
"This is not
a legal String."
"This is not a "legal" String either."
```

- A string can represent characters by preceding them with a backslash.
  - `\t` tab character
  - `\n` new line character
  - `\"` quotation mark character
  - `\\` backslash character
  - Example: `"Hello\tthere\nHow are you?"`

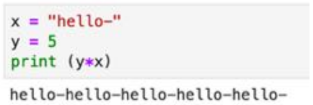
Strings and numerics



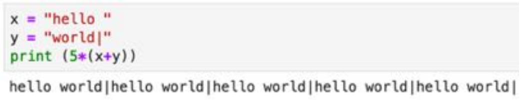
```
[2]: x = "hello"
     y = 5
     print (x+y)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-a7595bc38d88> in <module>
      1 x = "hello"
      2 y = 5
----> 3 print (x+y)

TypeError: can only concatenate str (not "int") to str

[ ]:
```



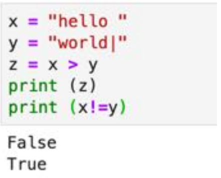
```
x = "hello"
y = "world"
print (x+y)

helloworld
```



```
x = "hello-"
y = 5
print (y*x)

hello-hello-hello-hello-hello-
```

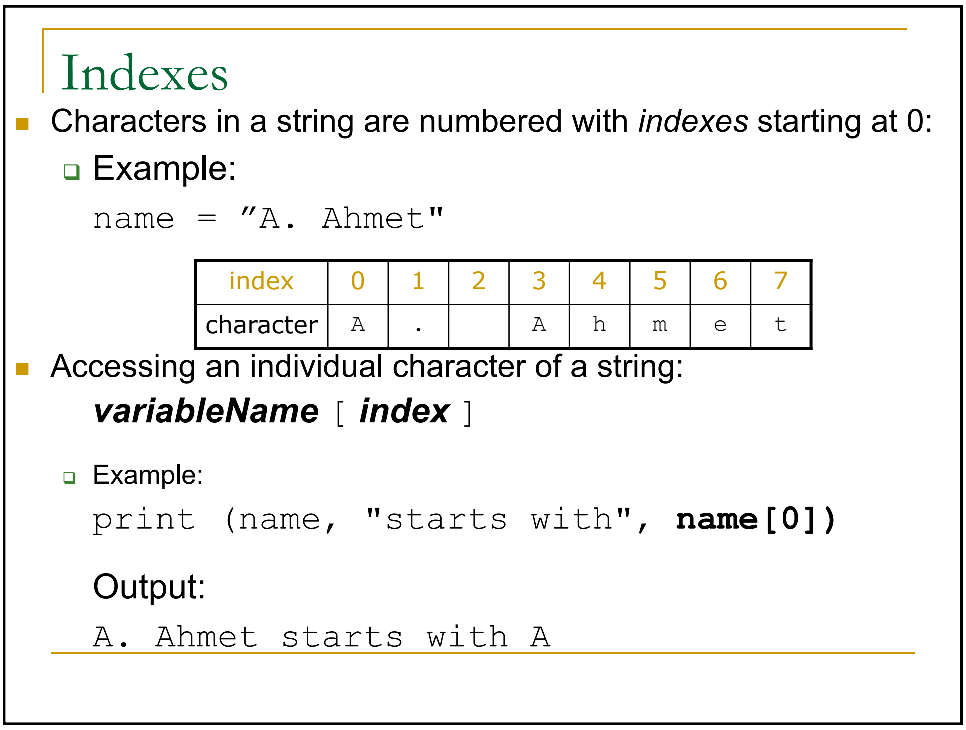


```
x = "hello "
y = "world|"
print (5*(x+y))

hello world|hello world|hello world|hello world|hello world|
```



```
x = "hello "
y = "world|"
z = x > y
print (z)
print (x!=y)

False
True
```

- Some **operators** apply to strings
  - + implies “concatenation”
  - * implies “multiple concatenation”
  - Relational operators (>, <, == …) can be used
- Python knows when it is dealing with a string or a number and behaves appropriately



Indexes

- Characters in a string are numbered with *indexes* starting at 0:
  - Example:

```
name = ”A. Ahmet"
```

index	0	1	2	3	4	5	6	7
character	A	.		A	h	m	e	t

- Accessing an individual character of a string:

  variableName [***index***]

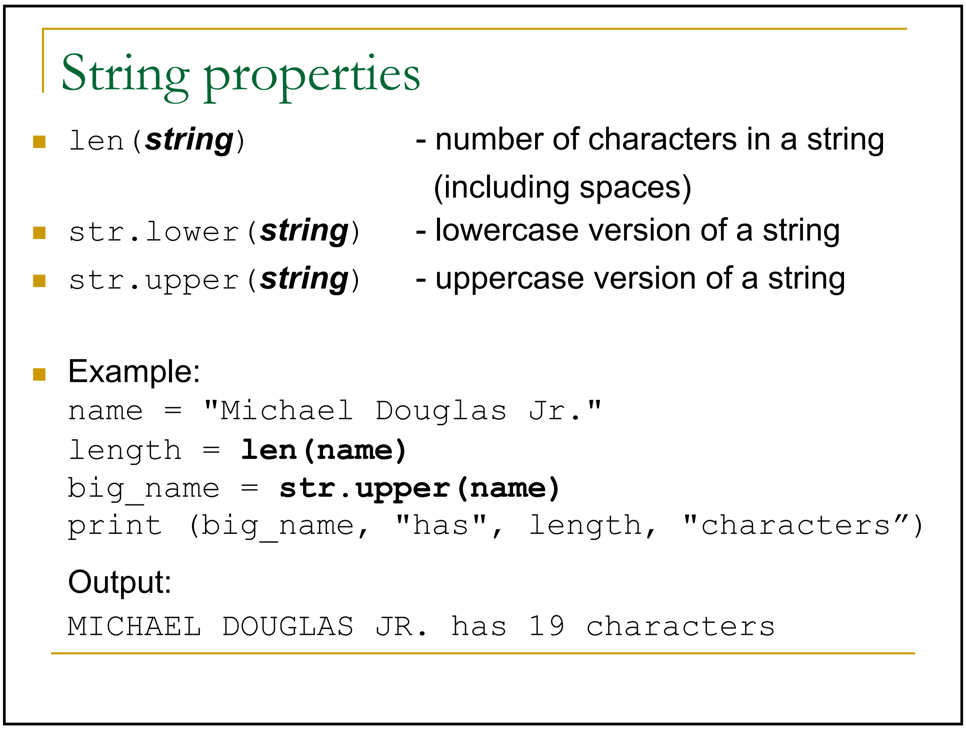
  - Example:

```
print (name, "starts with", name[0])
```

Output:

```
A. Ahmet starts with A
```

61



String properties

- `len(`***string***`)` - number of characters in a string (including spaces)
- `str.lower(`***string***`)` - lowercase version of a string
- `str.upper(`***string***`)` - uppercase version of a string

- Example:

```
name = "Michael Douglas Jr."
length = len(name)
big_name = str.upper(name)
print (big_name, "has", length, "characters”)
```

Output:

```
MICHAEL DOUGLAS JR. has 19 characters
```

62

Text processing

- **text processing**: Examining, editing, formatting text.
  - often uses loops that examine the characters of a string one by one
- A `for` loop can examine each character in a string in sequence.
  - Example:

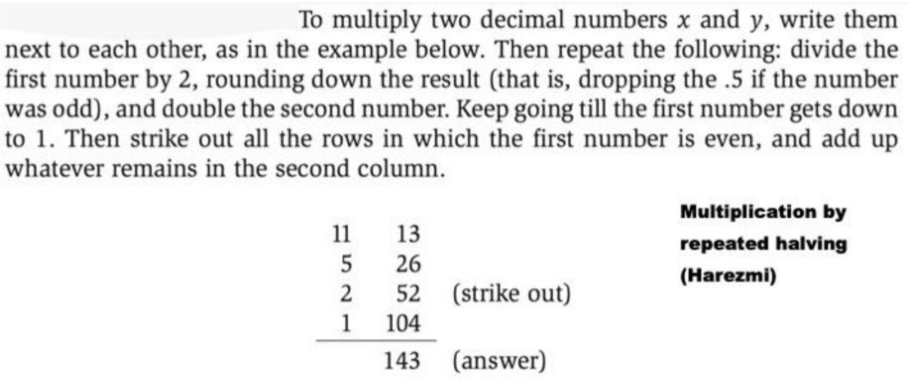
```
for c in ”hello":
    print (c)
```

Output:

```
h
e
l
l
o
```

Home study : Warm-up exercise to recall your basic programming skills (optional)

Al-Khwarizmi (Harezmi) Approach to easy multiplication



To multiply two decimal numbers x and y, write them next to each other, as in the example below. Then repeat the following: divide the first number by 2, rounding down the result (that is, dropping the .5 if the number was odd), and double the second number. Keep going till the first number gets down to 1. Then strike out all the rows in which the first number is even, and add up whatever remains in the second column.

11	13	
5	26	
2	52	(strike out)
1	104	
	143	(answer)

Multiplication by repeated halving (Harezmi)

Develop a Python program to multiply two numbers taken from the console input by using Khwarizmi algorithm. Result will be outputted to the console.

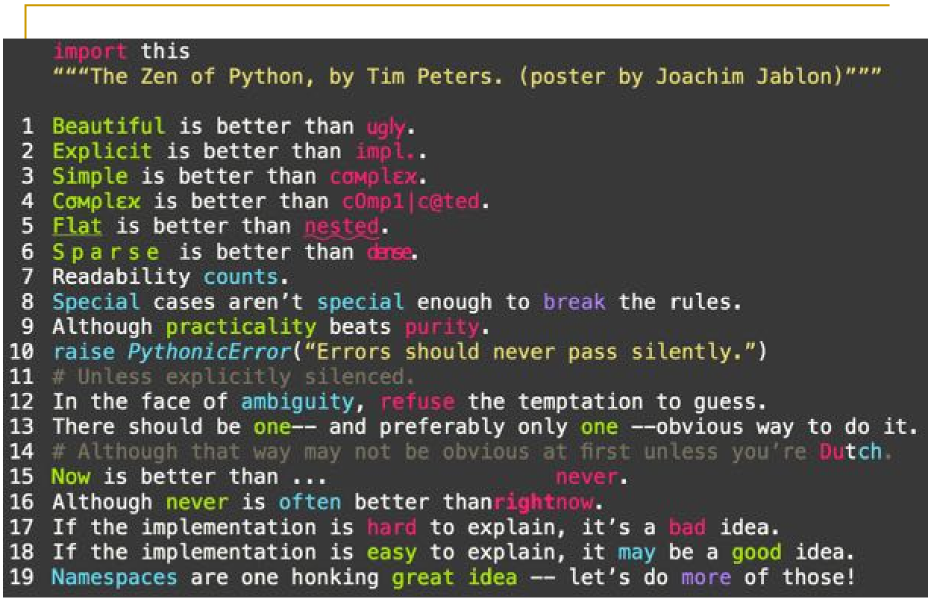
Home study

- In Jupyter, it is very common to use a special language to create text explanations to the code. It is called MARKDOWN.
- Your task is to research "Markdown" language and learn the basic uses.
  - Coloring
  - Text/content formatting
  - Writing formulas

Markdown is a lightweight, easy to learn markup language for formatting plain text. Its syntax has a one-to-one correspondance with HTML tags, so some prior knowledge would be helpful but is definitely not a prerequisite.

Ref: https://www.dataquest.io/blog/jupyter-notebook-tutorial/

65



```
import this
"""The Zen of Python, by Tim Peters. (poster by Joachim Jablon)"""

Beautiful is better than ugly.
Explicit is better than impl..
Simple is better than complex.
Complex is better than cOmp1|c@ted.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
raise PythonicError("Errors should never pass silently.")
# Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
# Although that way may not be obvious at first unless you're Dutch.
Now is better than ...                    never.
Although never is often better thanrightnow.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

66

SEE YOU NEXT WEEK!!!

DR. ORHAN GÖKÇÖL
gokcol@gmail.com
orhan.gokcol@ozyegin.edu.tr

67