PSA UNIVERSITY

REPUBLIC POLYTECHNIC
DISCOVER. TRANSFORM. ACHIEVE

Introductory Programming in Python

Day 1

python™

Introduction of trainer

Name
Seow Khee Wei

Phone
98463112

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Bachelor

BEng major in computer engineering (Nanyang Technological University)

Master

MBA
University of Melbourne

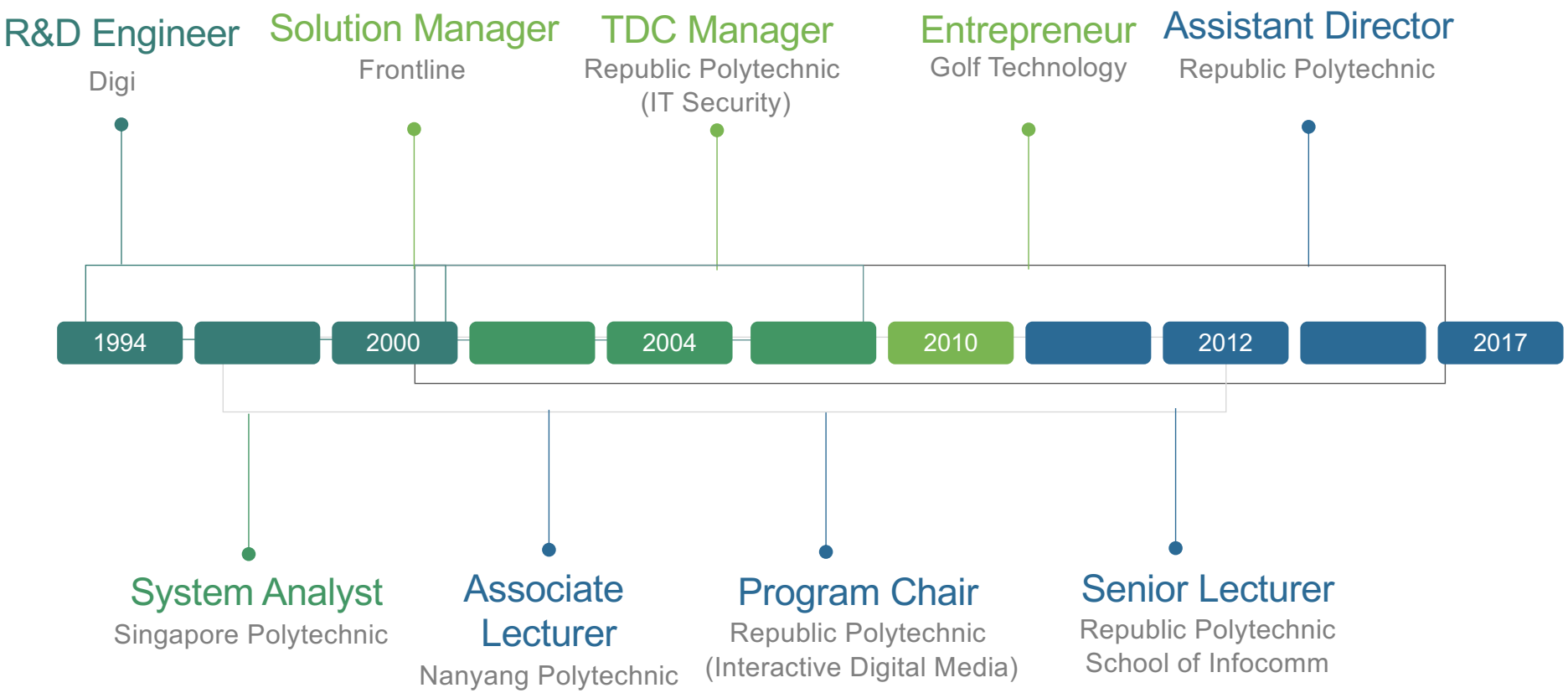
Post Graduate Certificate

Network Engineering (NTU)

Nano Degree

Artificial Intelligence

Introduction of trainer



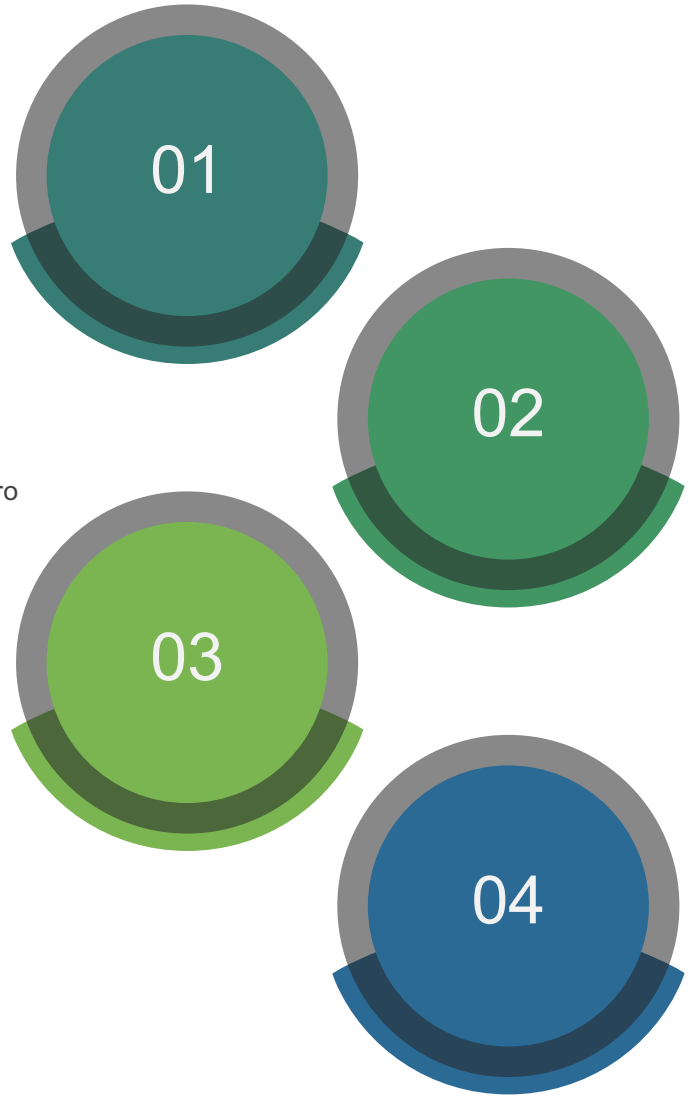
Portfolio

Technology Development Centre
Cognitive Systems Technology Centre
Data Analytics Technology Centre
IoT Solutions Centre

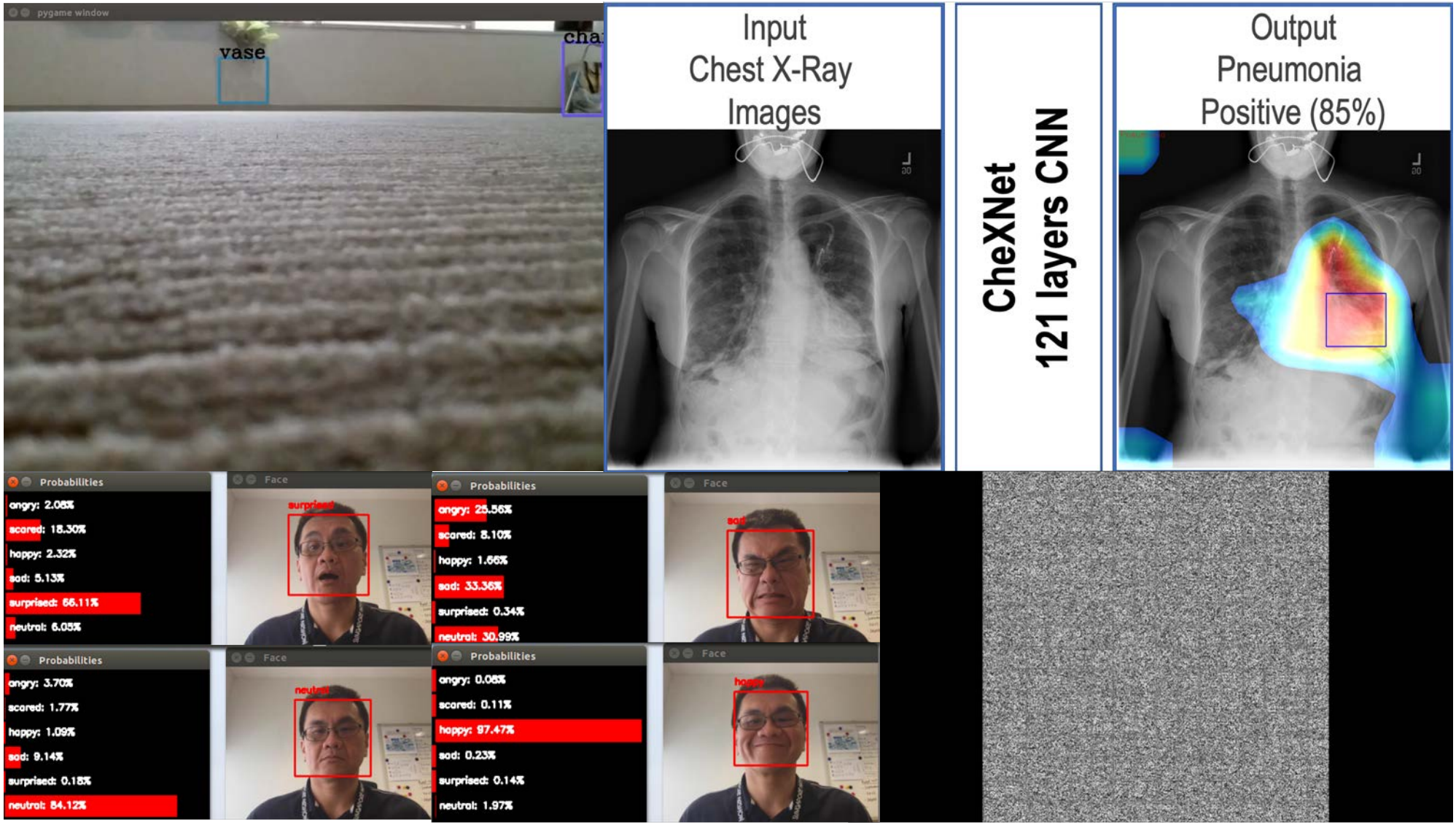
Industry Joint-Lab
Microsoft, Samsung, Element14, Starhub, RSA,
Palo Alto Network, UofG, Secura, Ixia, Trend Micro

Industry Engagement
Student Internship, FYP
Staff Attachment, Consultancy Projects

Training
Iphone Development, Programming, Artificial Intelligence



Pet Project


pygame window
vase
Input
Chest X-Ray
Images
CheXNet
121 layers CNN
Output
Pneumonia
Positive (85%)
Probabilities
angry: 2.08%
scared: 18.30%
happy: 2.32%
sad: 5.13%
surprised: 66.11%
neutral: 6.05%
Face
surprised
Probabilities
angry: 25.56%
scared: 8.10%
happy: 1.66%
sad: 33.36%
surprised: 0.34%
neutral: 30.99%
Face
sad
Probabilities
angry: 3.70%
scared: 1.77%
happy: 1.09%
sad: 9.14%
surprised: 0.15%
neutral: 84.12%
Face
neutral
Probabilities
angry: 0.06%
scared: 0.11%
happy: 97.47%
sad: 0.23%
surprised: 0.14%
neutral: 1.97%
Face
happy

What skill or talent do you wish you possessed?

When poll is active, respond at **PollEv.com/kheeweiseow794**

Text **KHEEWEISEOW794** to **+65 8241 0042** once to join

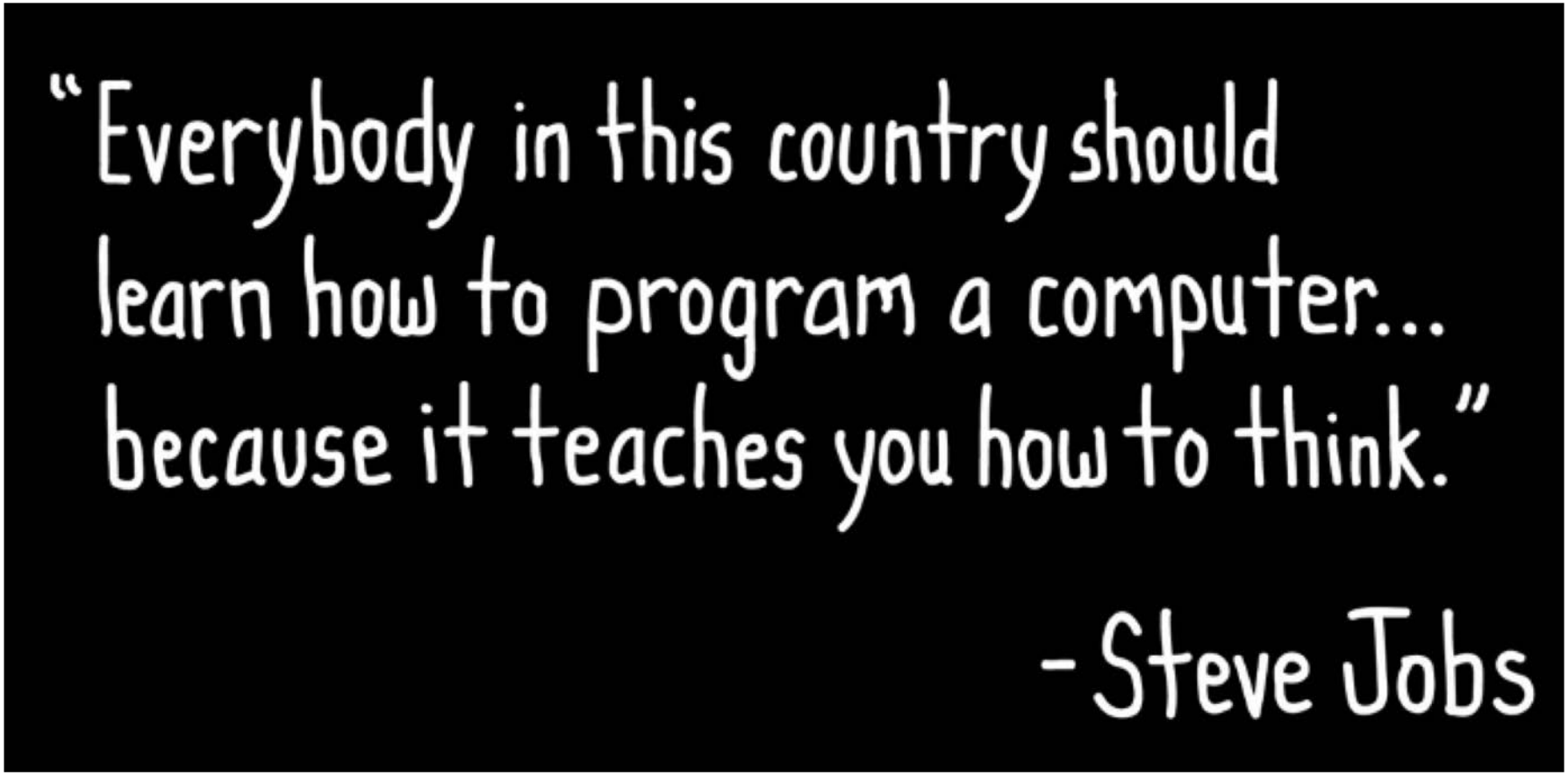
Background

When survey is active, respond at **PollEv.com/kheeweiseow794**

0 surveys done

0 surveys underway

Why are we here?


"Everybody in this country should
learn how to program a computer...
because it teaches you how to think."
- Steve Jobs

About this workshop

- Learn about Python 3, a very versatile and useful language
- Discuss its advantages and disadvantages (also what to look out for)
- Improve your problem solving skills:
  How to automate the most boring and repetitive stuff using Python
- Some tools and useful modules you can use to build your applications

Primary Focus

- This course is an introduction to Python Programming
- Designed for people seeking to improve their efficiency at work (and life!)
- This course merges two topics:
  Learning Python + becoming a problem solver using code
- You will learn Python through solving problems:
  Learn by doing!

Approach

- Coding coding coding (Steve Jobs)
- Learn as we need it (and how to be resourceful)
- Ask questions and for help when you get stuck.
- Don’t be afraid to try new stuff (and make mistakes)
- How to use (aka read) the documentation

Prereqs and preparations

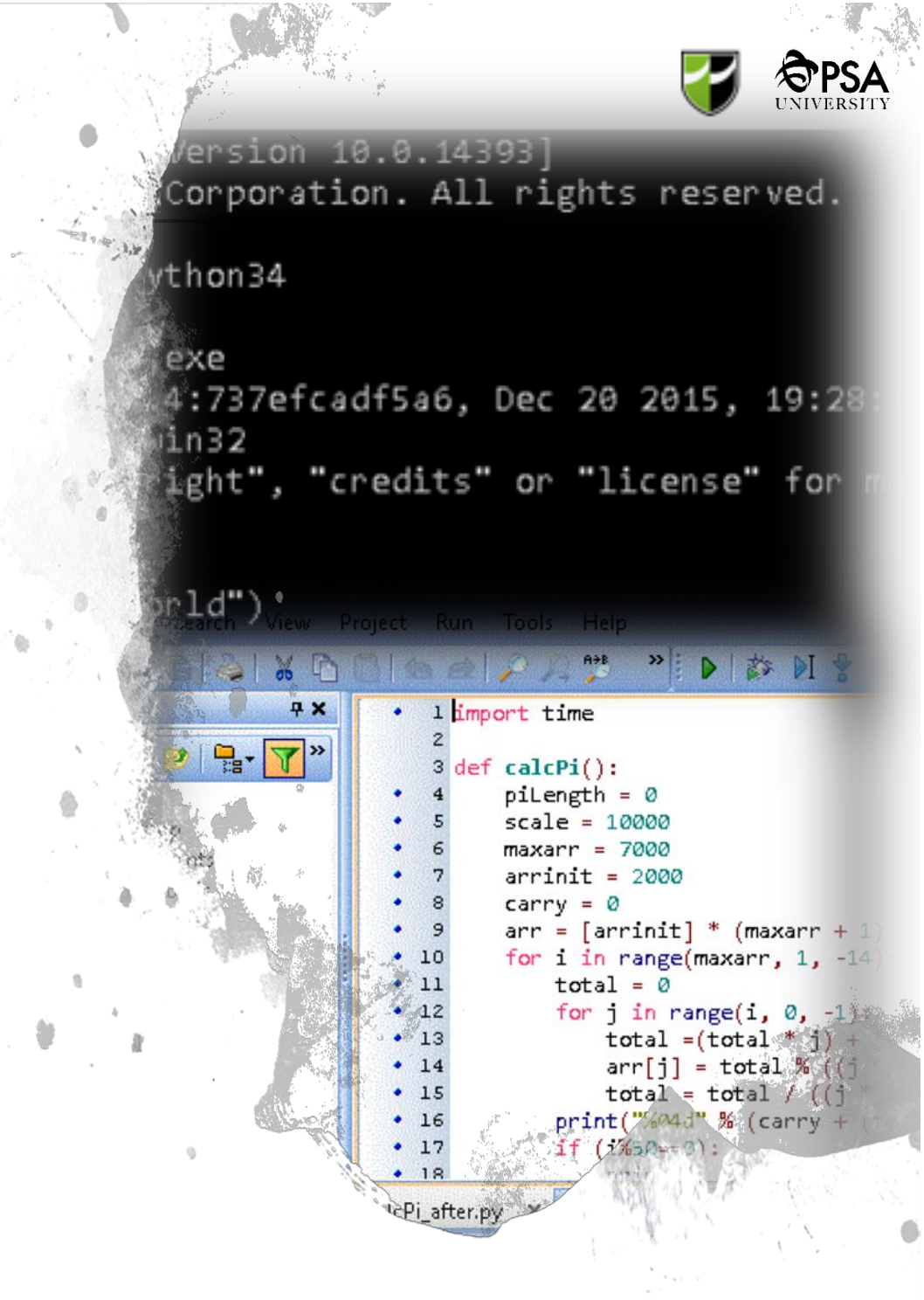
Before you attend this workshop, please make sure:

- Your laptop works

- You have installed Anaconda 5.3.0 for Python 3.7
- You installed a decent editor, We are using PyCharm Community Edition in this course
- You should (preferably) have some basic programming experience and be familiar with basic concepts such as variables, flow control and functions

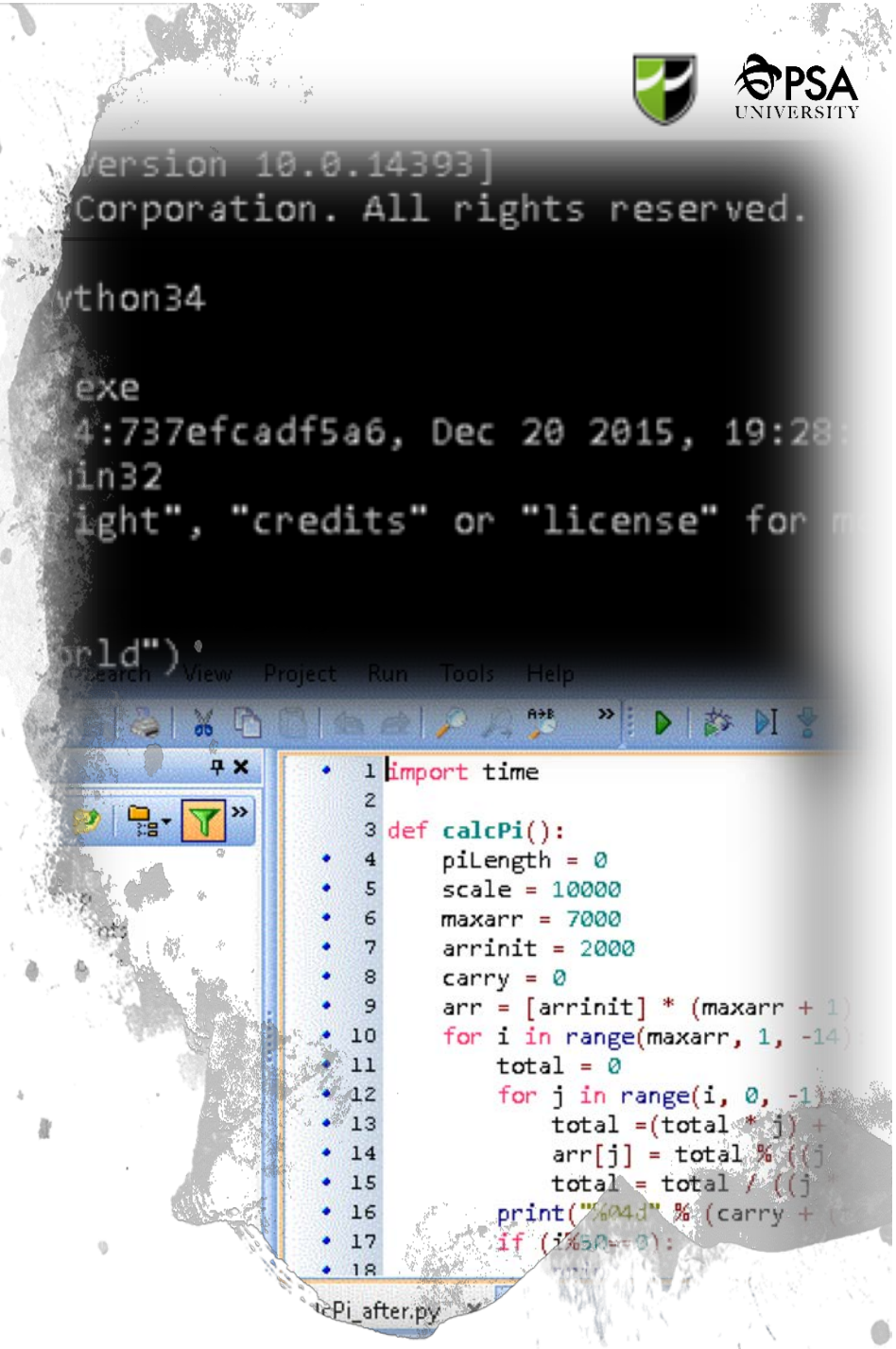
Programme Day One

- A brief history of Python
- Setting up python environment
- Learn the basics:
  - Data types
  - Conversions
  - String operations
  - Functions
  - And more!
- String functions, formatting
- Find files by name, by extension, by size, by content and calculate the total size



Programme
Day Two

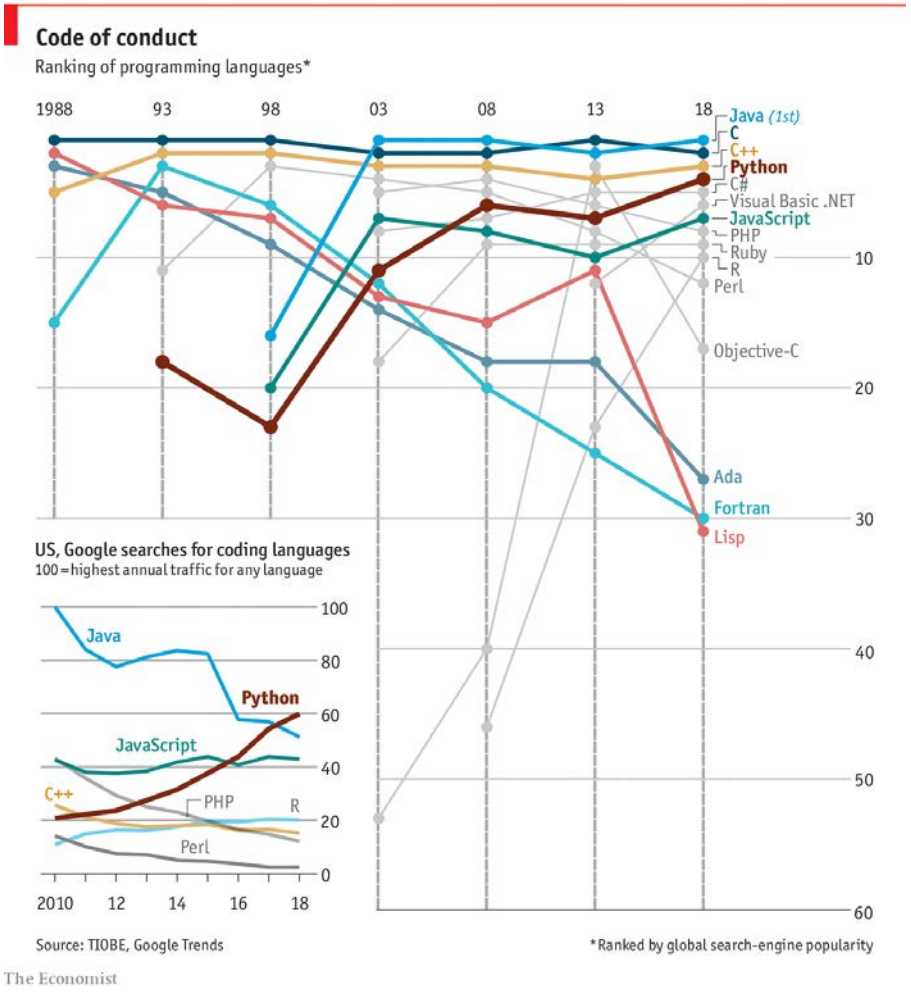
- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Processing CSV files
- Image processing: loading, scaling, watermark, applying filters
- Connecting to the Web
- Sending emails
- Graphical User Interface
- AI – Facial Recognition



Introduction to Python

What is Python?

- Interpreted
- Interactive
- Functional
- Object-oriented
- Programming language, not just a scripting language



Introduction to Python

- Allows modular programming
- Great emphasis on readability:
  - Codes are forced to be indented
  - But there's a practical drawback to this
- Easy to embed in and extend with other languages
- Easy to learn for beginners
- Completely FREE!
- Copyrighted but use not restricted

Introduction to Python

- Interfaces to:
  - COM, DCOM, ODBC
  - Most databases (MySQL, MSQL, Sybase, Oracle,…)
  - Java (Jpython)
  - Many GUI / GFX libraries
    - Platform-independent: Tk, wxWindows, GTK, Pyglet
    - Platform-specific: MFC, MacOS, X11

The Hitchhiker's Guide to Python! http://bit.ly/2T3gzfm

Introduction to Python

Python vs. Perl:
- Easier to learn
- More readable
- Fewer side effects
- Less Unix bias

Python vs. Tcl:
- Much faster
- Less need for C extensions
- Better Java integration

Python vs. Java:
- More concise code
- Dynamic typing
- Runs slower but development is fast
- No native-code compilation
- Can be integrated with Java using JPython

Nov 2018	Nov 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.746%	+3.51%
2	2		C	14.396%	+5.10%
3	3		C++	8.282%	+2.94%
4	4		Python	7.683%	+3.20%
5	7	^	Visual Basic .NET	6.490%	+3.58%
6	5	v	C#	3.952%	+0.94%
7	6	v	JavaScript	2.655%	-0.32%
8	8		PHP	2.376%	+0.48%
9	-	^^	SQL	1.844%	+1.84%
10	14	^^	Go	1.495%	-0.07%
11	19	^^	Objective-C	1.476%	+0.06%
12	20	^^	Swift	1.455%	+0.07%
13	9	vv	Delphi/Object Pascal	1.423%	-0.32%
14	11	v	R	1.407%	-0.20%
15	10	vv	Assembly language	1.108%	-0.61%
16	13	v	Ruby	1.091%	-0.50%
17	12	vv	MATLAB	1.030%	-0.57%
18	15	v	Perl	1.001%	-0.56%
19	18	v	PL/SQL	1.000%	-0.45%
20	17	v	Visual Basic	0.854%	-0.63%

https://www.tiobe.com/tiobe-index/

Introduction to Python

Who uses Python?

01 ***Web Development.*** Google (in search spiders), Yahoo (in maps applcaitions)

02 ***Games.*** Civilization 4 (game logic & AI), Battlefield 2 (score keeping and team balancing)

03 ***Graphics***. Industrial Light & Magic (rendering), Blender 3D (extension Language)

04 ***Financial.*** ABN AMRO Bank (communicate trade information between systems.

05 ***Science.*** National Weather Centre, US (make maps, create forecasts, etc) NASA (Integrated Planning System)

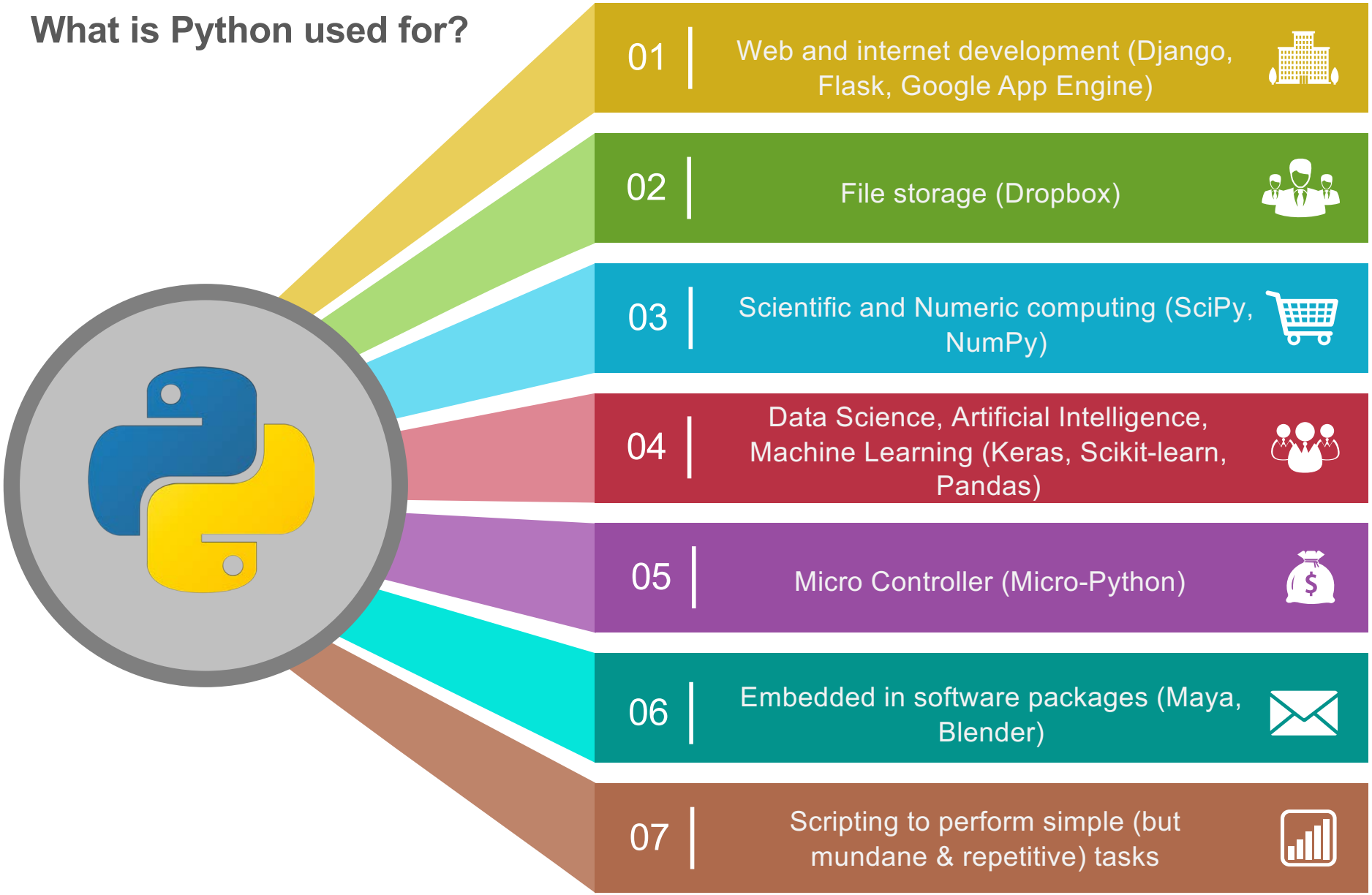
06 ***Education.*** University of California, Irvine. University of New South Wales (Australia), NUS, SUTD, SMU, RP

07 ***Business Software.*** Thawte Consulting, IBM, RealNetowkrs

More examples: http://wiki.python.org/moin/OrganizationsUsingPython

Introduction to Python

What is Python used for?

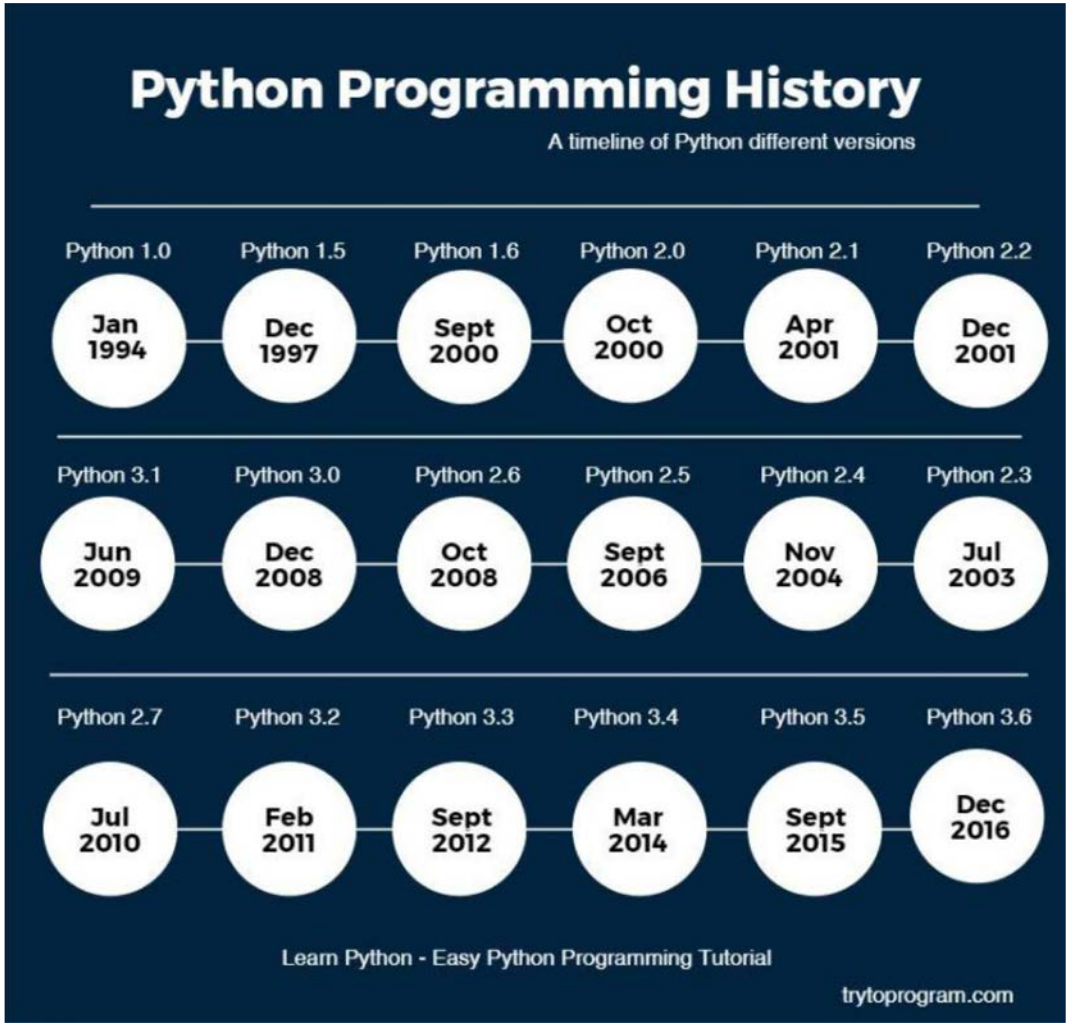


Introduction to Python

Why the name, Python?

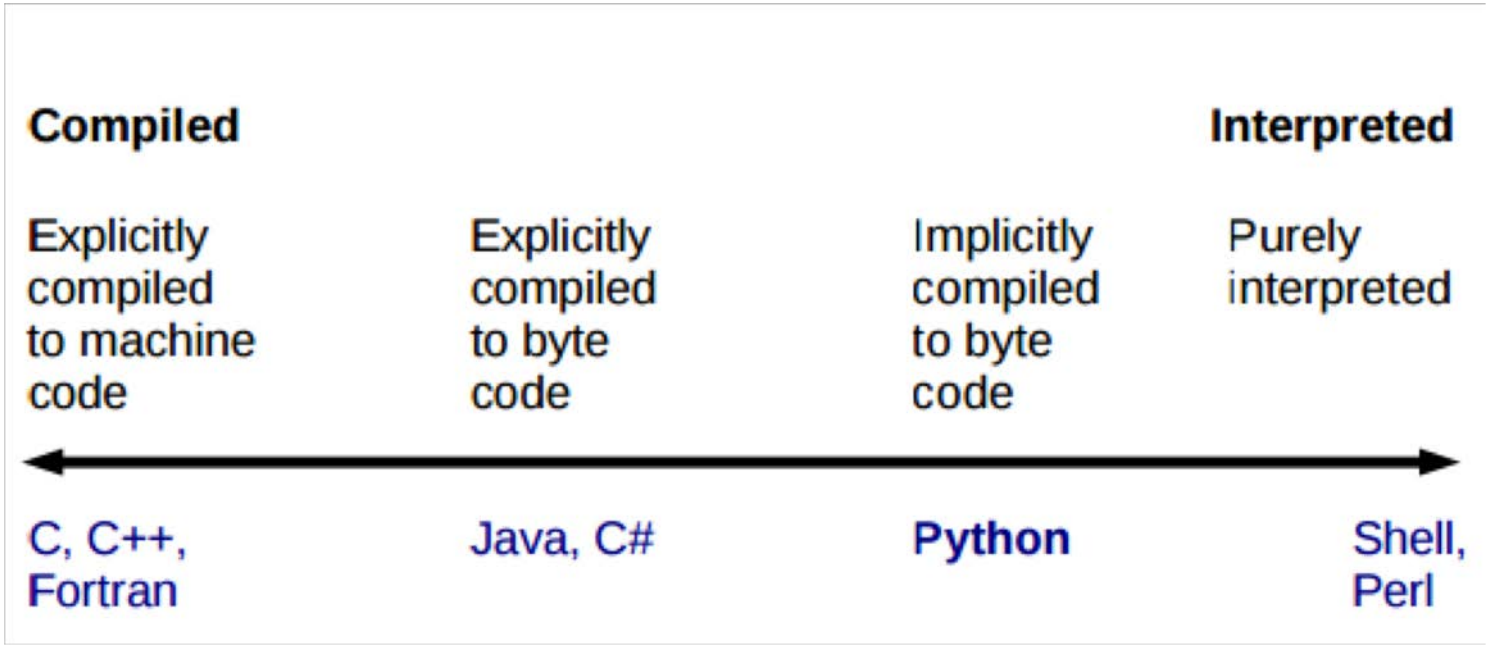
- Originally not a snake, but from the British comedy "**Monty Python's Flying Circus**". The snake logo came later.
- Invented in 1990 by Guido Van Rossum
- Initially intended to be a scripting language on Amoeba OS
- Python was influenced by ABC and Modula-3
- First public release was in 1991

Introduction to Python



Python has two versions currently: **2.7.14 and 3.7**

Where is Python on the Map



Python 2 vs. Python 3

- **Different syntax**: e.g. print statement, division
  - Python 2
    - ✓ print “Hello World!”
    - ✓ x = 5 / 2 # x’s value will be 2
  - Python 3
    - ✓ print(“Hello World!”) # brackets are compulsory now
    - ✓ x = 5 / 2 # x’s value will be 2.5
- **Which to learn?**
  - Many major frameworks and third-party modules have already migrated or are in the process of moving to Python 3
  - Python 2’s EOL is in 2020, no Python 2.8
  - **The obvious pick: Python 3**

Getting started

Install and setup

- Get it @ https://www.anaconda.com/download/
- Use Python 3.7 for this course
- Set up virtual environments

How to write and run Python programs

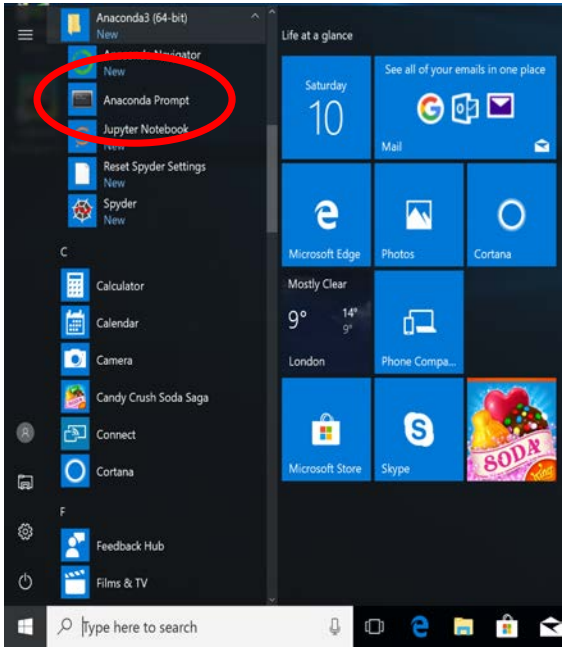
- Use a text editor (notepad, vi, emacs, etc)
- Use an IDE (IDLE, PyScripter, PyCharm, etc)

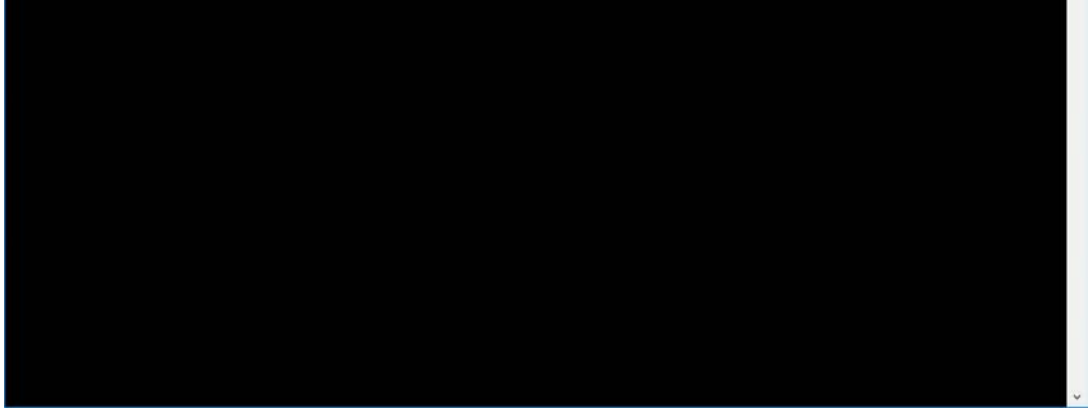
Experiment with Jupyter Notebook

- Installed together with Anaconda

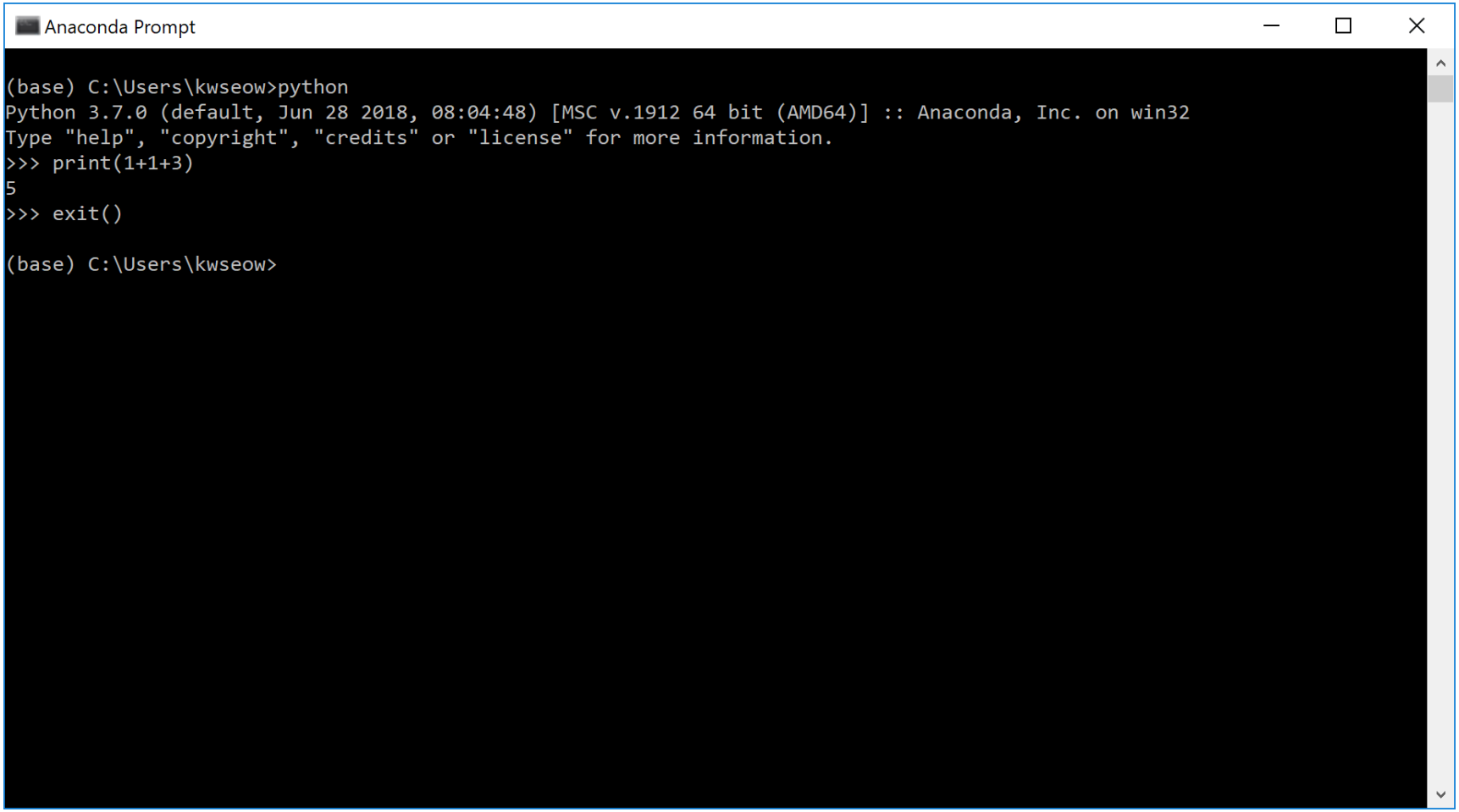
Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users and includes more than 250 popular data-science packages suitable for Windows, Linux, and MacOS.

Running Python the basic way

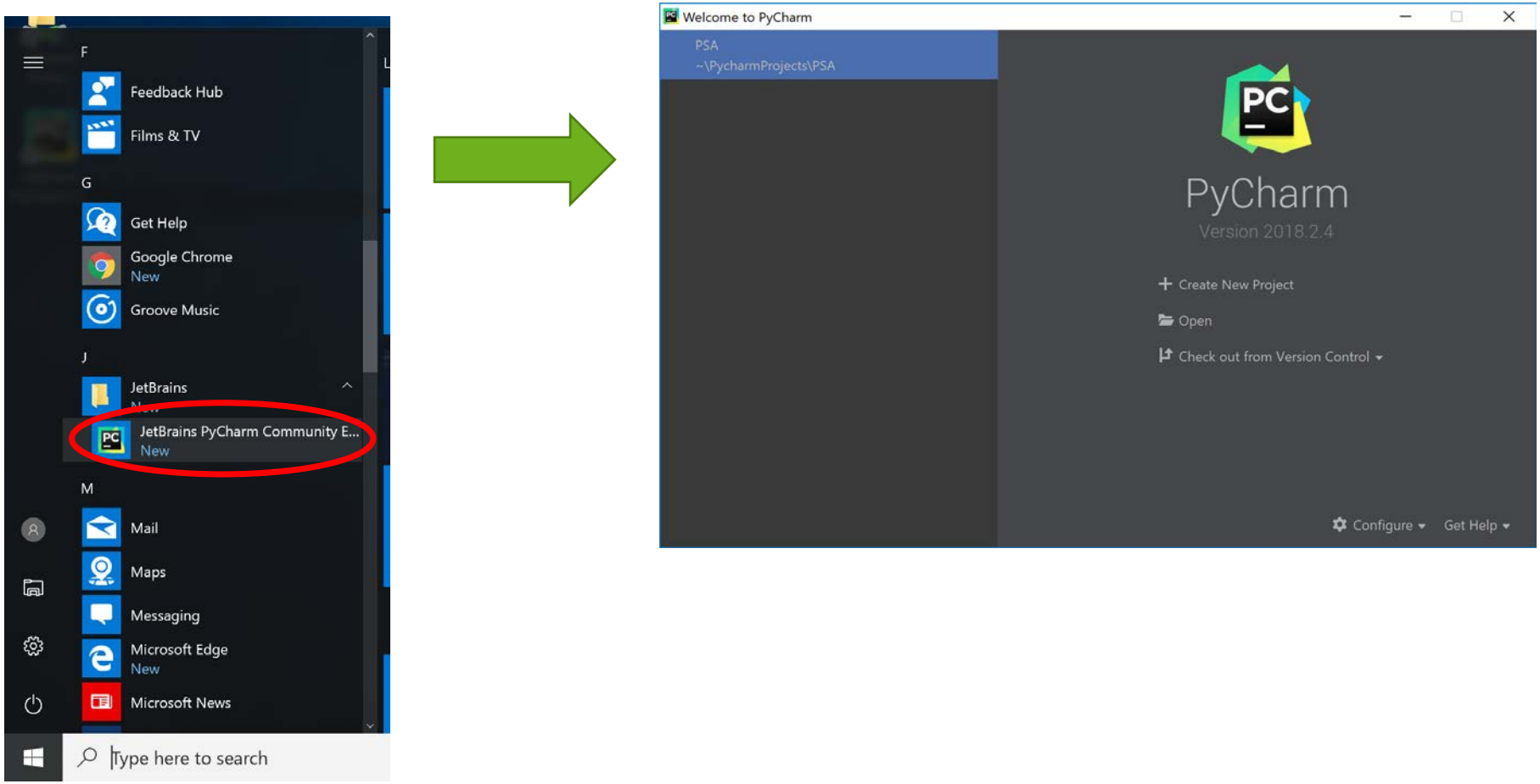




First, Experience the Basic Way



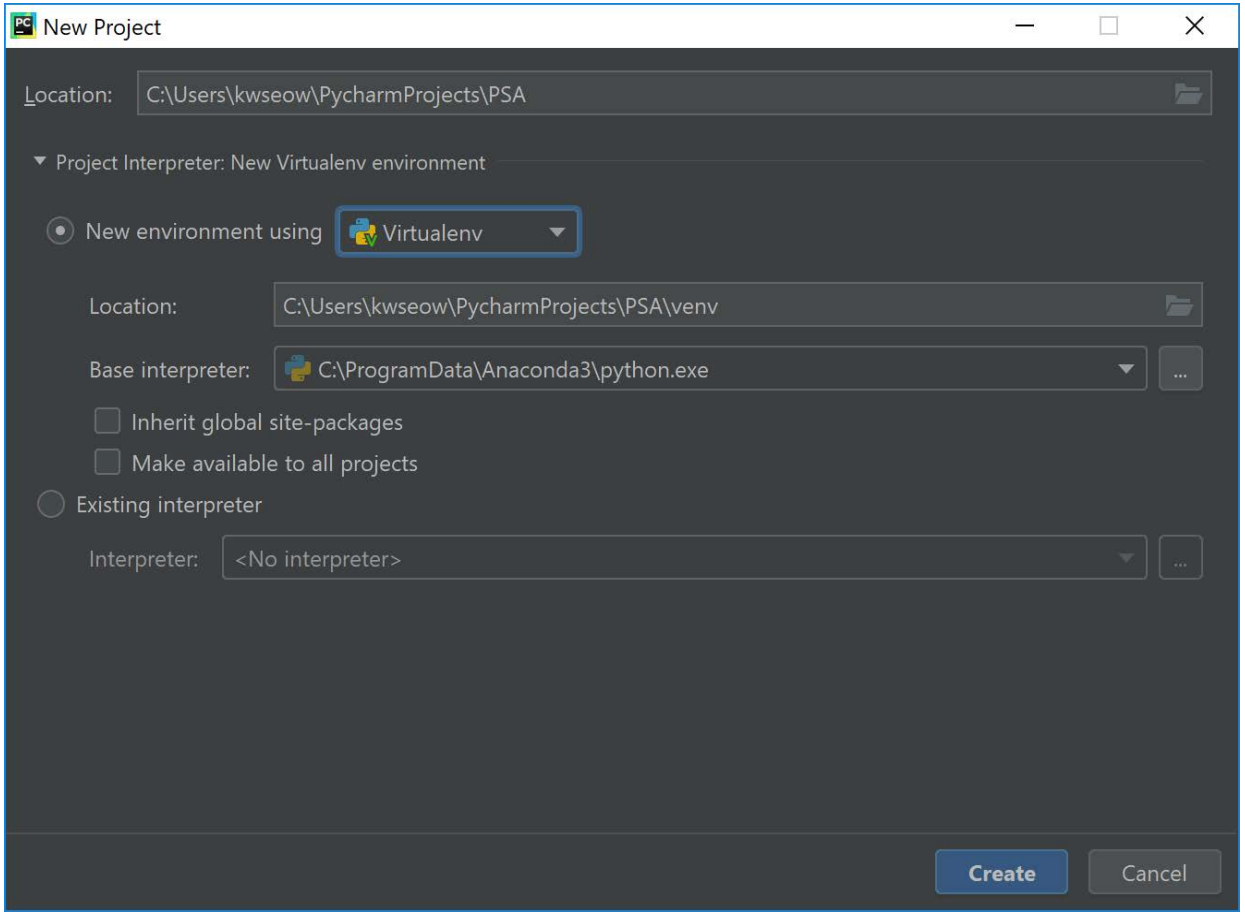
Use an IDE, Run PyCharm



Use an IDE, Run PyCharm

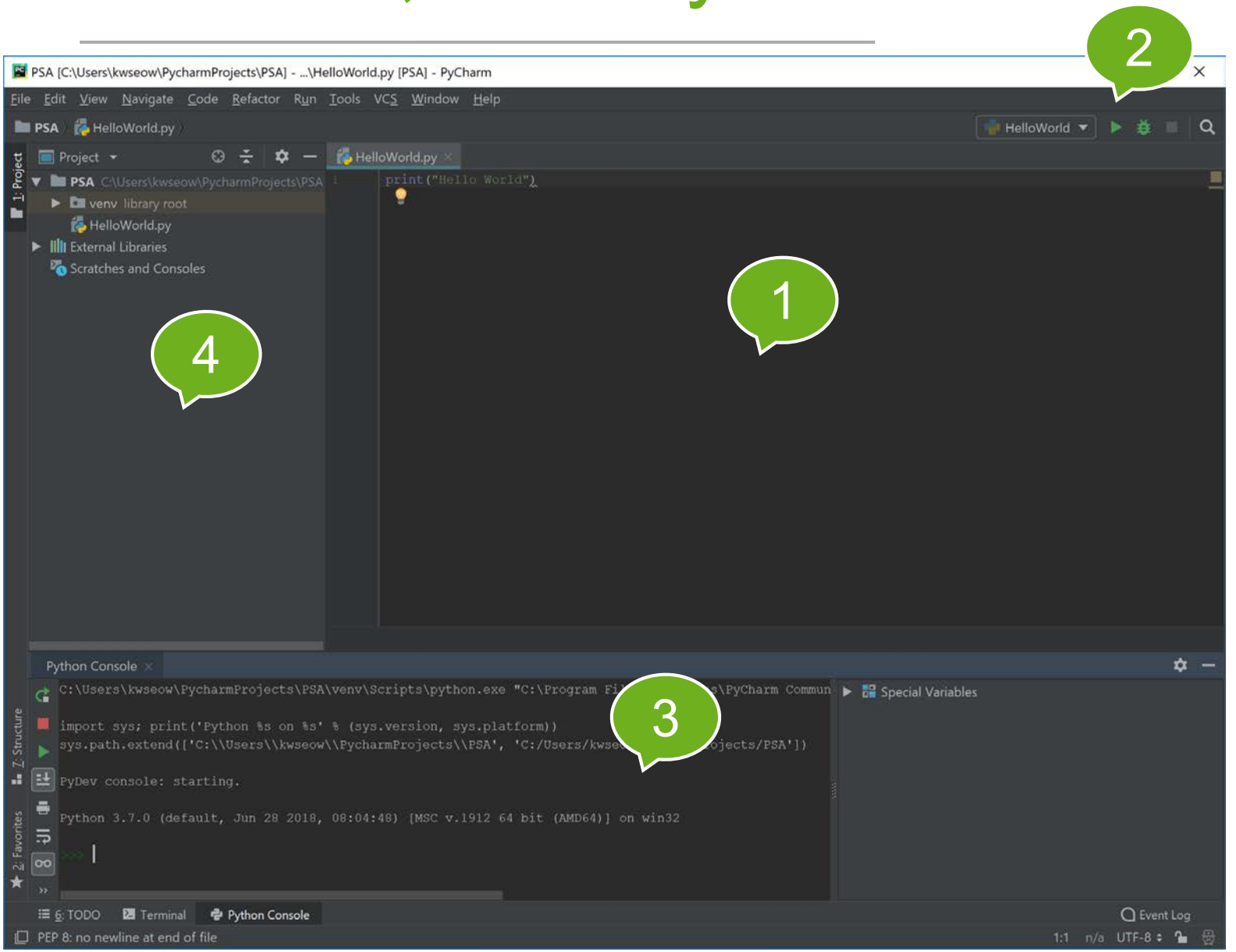
- Enter the project info

- Use the system installed Python interpreter



Use an IDE, Run PyCharm

1. Editor
2. Run button
3. Output window / Console
4. Your files



PyCharm

DEFAULT KEYMAP

Editing

Ctrl + Space	Basic code completion (the name of any class, method or variable)
Ctrl + Alt + Space	Class name completion (the name of any project class independently of current imports)
Ctrl + Shift + Enter	Complete statement
Ctrl + P	Parameter info (within method call arguments)
Ctrl + Q	Quick documentation lookup
Shift + F1	External Doc
Ctrl + mouse over code	Brief Info
Ctrl + F1	Show descriptions of error or warning at caret
Alt + Insert	Generate code...
Ctrl + O	Override methods
Ctrl + Alt + T	Surround with...
Ctrl + /	Comment/uncomment with line comment
Ctrl + Shift + /	Comment/uncomment with block comment
Ctrl + W	Select successively increasing code blocks
Ctrl + Shift + W	Decrease current selection to previous state
Ctrl + Shift +]	Select till code block end
Ctrl + Shift + [	Select till code block start
Alt + Enter	Show intention actions and quick-fixes
Ctrl + Alt + L	Reformat code
Ctrl + Alt + O	Optimize imports
Ctrl + Alt + I	Auto-indent line(s)
Tab	Indent selected lines
Shift + Tab	Unindent selected lines
Ctrl + X , Shift + Delete	Cut current line or selected block to clipboard
Ctrl + C , Ctrl + Insert	Copy current line or selected block to clipboard
Ctrl + V , Shift + Insert	Paste from clipboard
Ctrl + Shift + V	Paste from recent buffers...
Ctrl + D	Duplicate current line or selected block
Ctrl + Y	Delete line at caret
Ctrl + Shift + J	Smart line join
Ctrl + Enter	Smart line split
Shift + Enter	Start new line
Ctrl + Shift + U	Toggle case for word at caret or selected block
Ctrl + Delete	Delete to word end
Ctrl + Backspace	Delete to word start
Ctrl + NumPad+	Expand code block
Ctrl + NumPad-	Collapse code block
Ctrl + Shift + NumPad+	Expand all
Ctrl + Shift + NumPad-	Collapse all
Ctrl + F4	Close active editor tab

Running

Alt + Shift + F10	Select configuration and run
Alt + Shift + F9	Select configuration and debug
Shift + F10	Run
Shift + F9	Debug
Ctrl + Shift + F10	Run context configuration from editor
Ctrl + Alt + R	Run manage.py task

Debugging

F8 / F7	Step over/into
Shift + F8	Step out
Alt + F9	Run to cursor
Alt + F8	Evaluate expression
Ctrl + Alt + F8	Quick evaluate expression
F9	Resume program
Ctrl + F8	Toggle breakpoint
Ctrl + Shift + F8	View breakpoints

Navigation

Ctrl + N	Go to class
Ctrl + Shift + N	Go to file
Ctrl + Alt + Shift + N	Go to symbol
Alt + Right	Go to next editor tab
Alt + Left	Go to previous editor tab
F12	Go back to previous tool window
Esc	Go to editor (from tool window)
Shift + Esc	Hide active or last active window
Ctrl + Shift + F4	Close active run/messages/find/... tab
Ctrl + G	Go to line
Ctrl + E	Recent files popup
Ctrl + Alt + Right	Navigate forward
Ctrl + Alt + Left	Navigate back
Ctrl + Shift + Backspace	Navigate to last edit location
Alt + F1	Select current file or symbol in any view
Ctrl + B , Ctrl + Click	Go to declaration
Ctrl + Alt + B	Go to implementation(s)
Ctrl + Shift + I	Open quick definition lookup
Ctrl + Shift + B	Go to type declaration
Ctrl + U	Go to super-method/super-class
Alt + Up / Down	Go to previous/next method
Ctrl +] / [	Move to code block end/start
Ctrl + F12	File structure popup
Ctrl + H	Type hierarchy
Ctrl + Shift + H	Method hierarchy
Ctrl + Alt + H	Call hierarchy
F2 / Shift + F2	Next/previous highlighted error
F4	Edit source
Ctrl + Enter	View source
Alt + Home	Show navigation bar
F11	Toggle bookmark
Ctrl + Shift + F11	Toggle bookmark with mnemonic
Ctrl + #[0-9]	Go to numbered bookmark
Shift + F11	Show bookmarks

Search/Replace

Ctrl + F / Ctrl + R	Find/Replace
F3 / Shift + F3	Find next/previous
Ctrl + Shift + F	Find in path
Ctrl + Shift + R	Replace in path

Usage Search

Alt + F7 / Ctrl + F7	Find usages / Find usages in file
Ctrl + Shift + F7	Highlight usages in file
Ctrl + Alt + F7	Show usages

Refactoring

F5 / F6	Copy / Move
Alt + Delete	Safe Delete
Shift + F6	Rename
Ctrl + F6	Change Signature
Ctrl + Alt + N	Inline
Ctrl + Alt + M	Extract Method
Ctrl + Alt + V	Extract Variable
Ctrl + Alt + F	Extract Field
Ctrl + Alt + C	Extract Constant
Ctrl + Alt + P	Extract Parameter

VCS/Local History

Ctrl + K	Commit project to VCS
Ctrl + T	Update project from VCS
Alt + Shift + C	View recent changes
Alt + BackQuote (`)	'VCS' quick popup

Live Templates

Ctrl + Alt + J	Surround with Live Template
Ctrl + J	Insert Live Template

General

Alt + #[0-9]	Open corresponding tool window
Ctrl + S	Save all
Ctrl + Alt + Y	Synchronize
Ctrl + Shift + F12	Toggle maximizing editor
Alt + Shift + F	Add to Favorites
Alt + Shift + I	Inspect current file with current profile
Ctrl + BackQuote (`)	Quick switch current scheme
Ctrl + Alt + S	Open Settings dialog
Ctrl + Shift + A	Find Action
Ctrl + Tab	Switch between tabs and tool window

To find any action inside the IDE use Find Action (Ctrl+Shift+A)

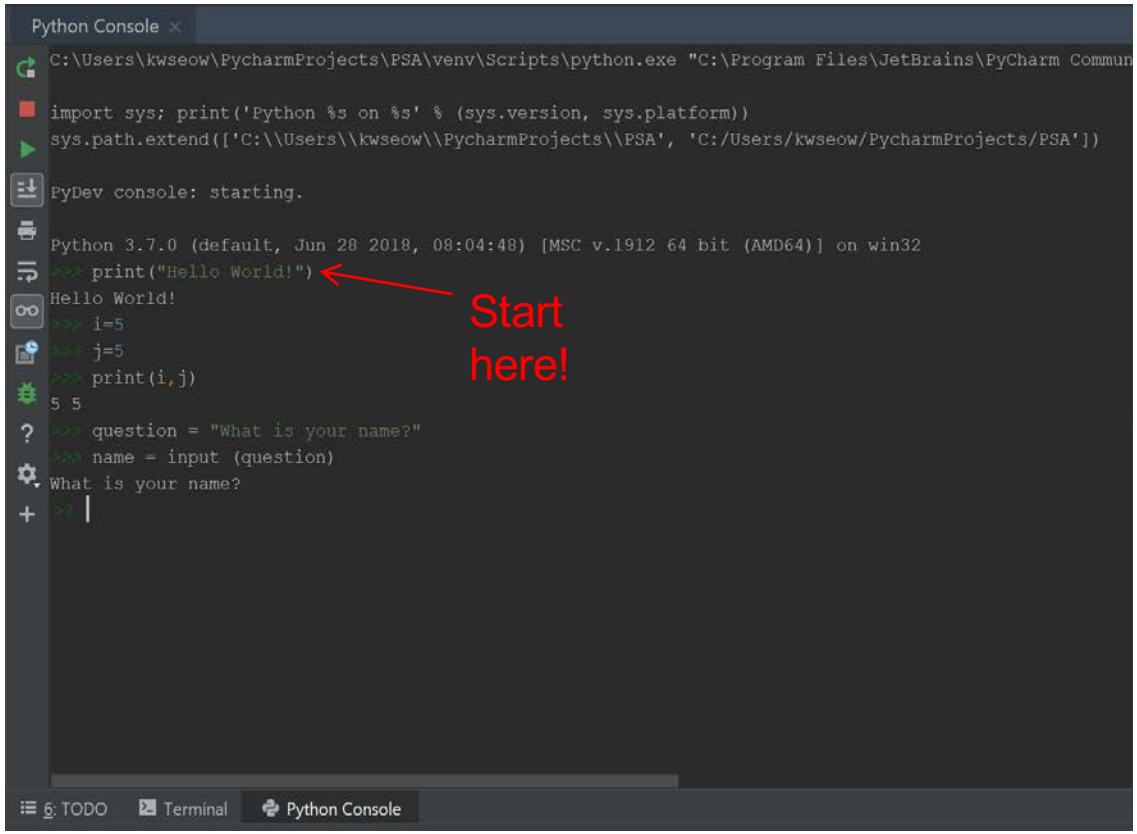

jetbrains.com/pycharm


blog.jetbrains.com/pycharm


@pycharm

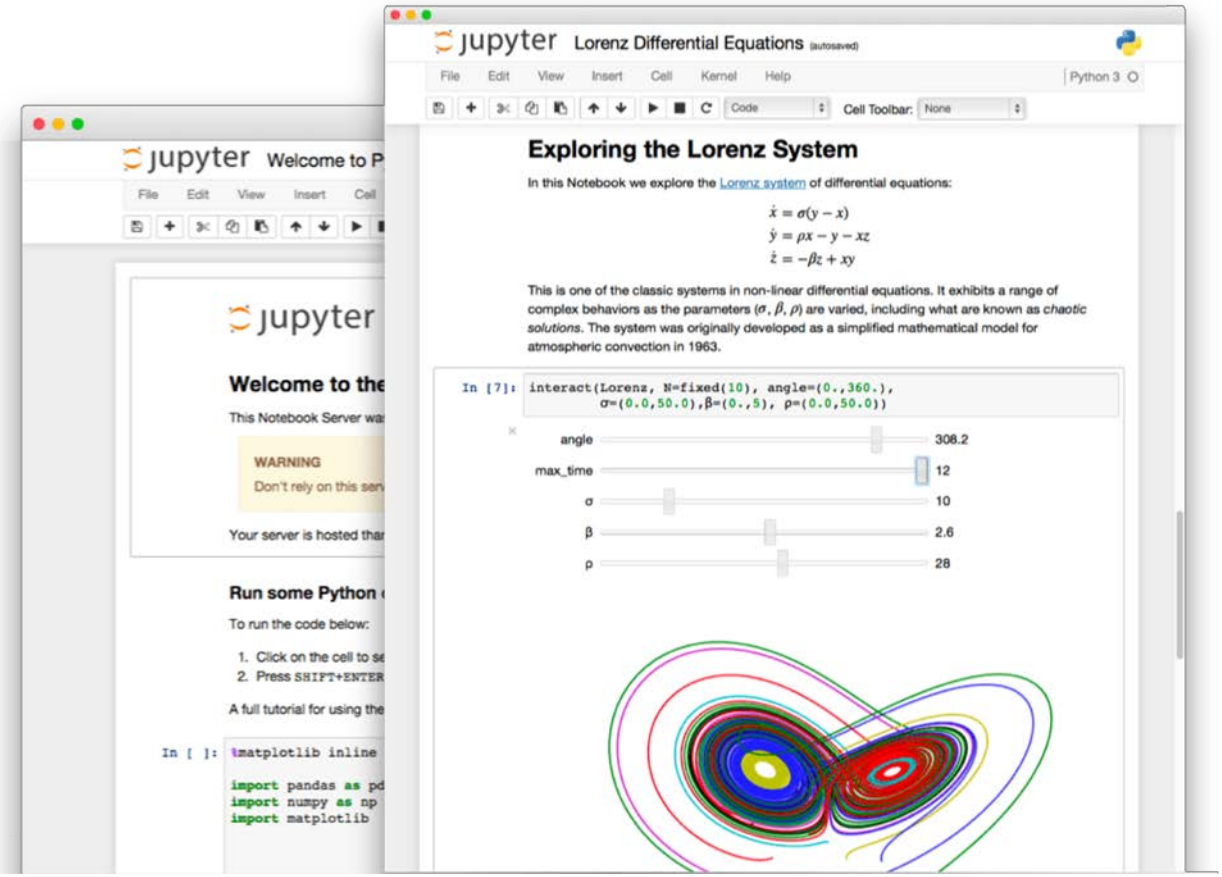
Using the console

- Also known as the interpreter
- See the output straightaway
- Usually used to test very small chunks of code
- Type code after >>>
- Let's try!



Using Jyputer Notebook

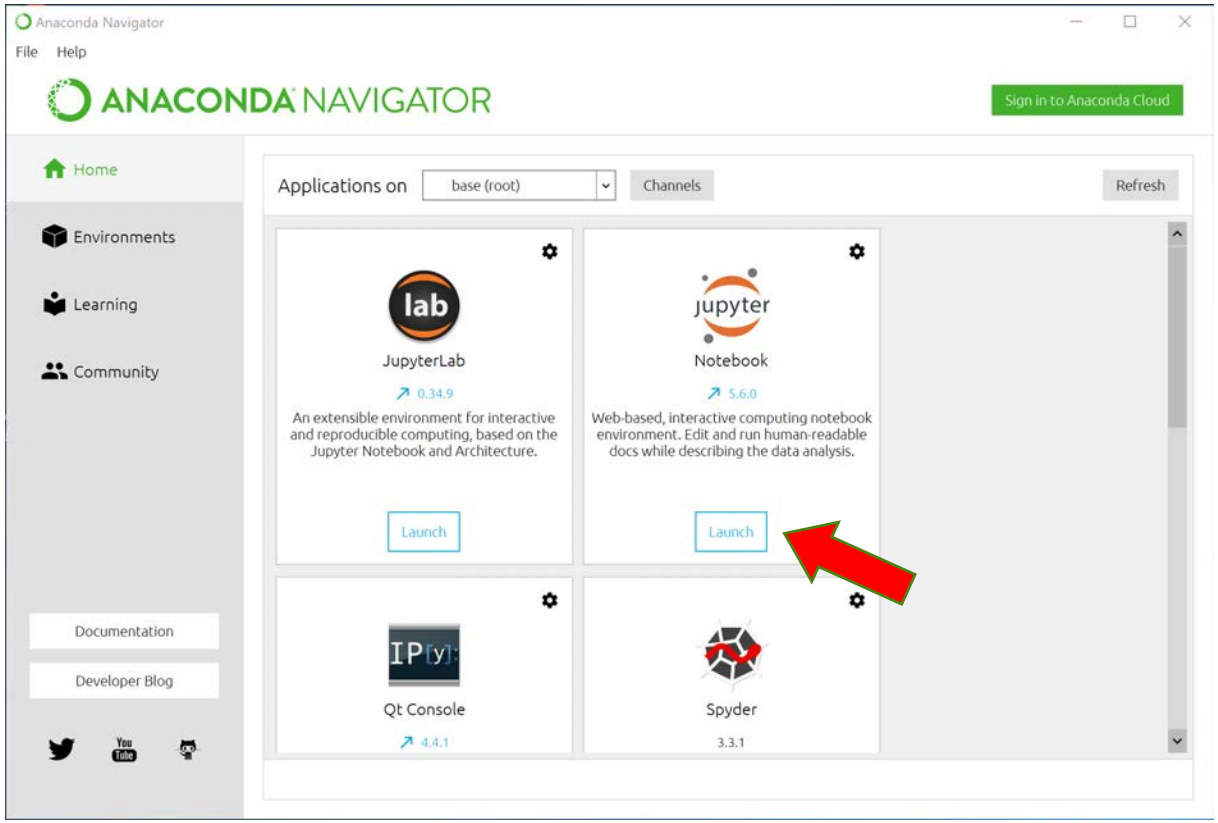
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



Using Jyputer Notebook

Launch Anaconda Navigator from the start menu

Click on Jupyter Notebook to launch Jupyter

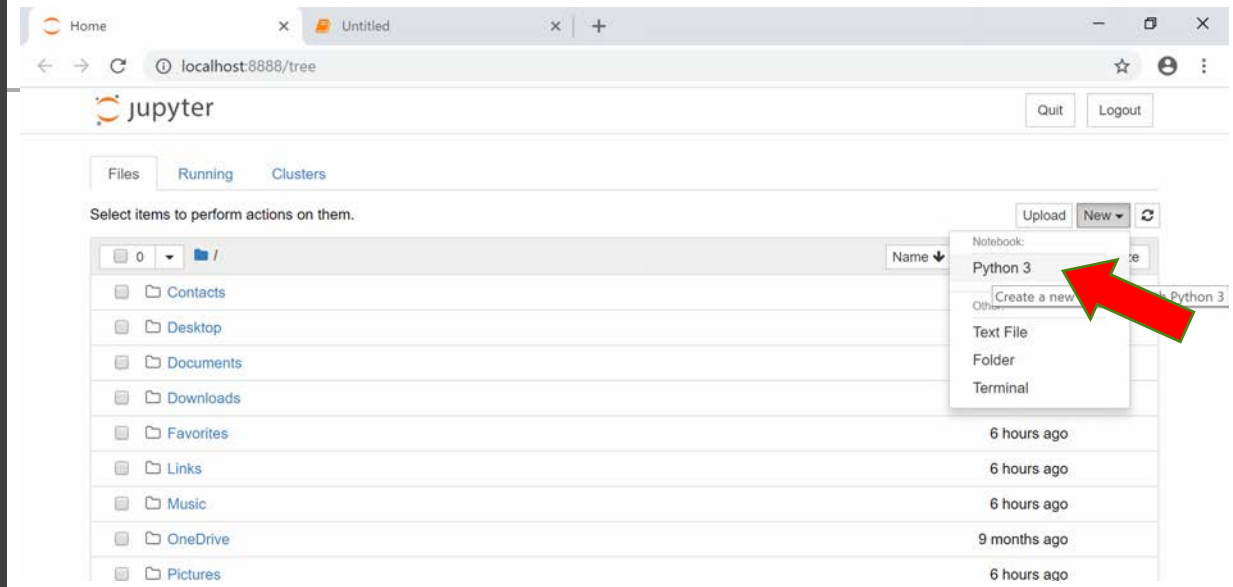


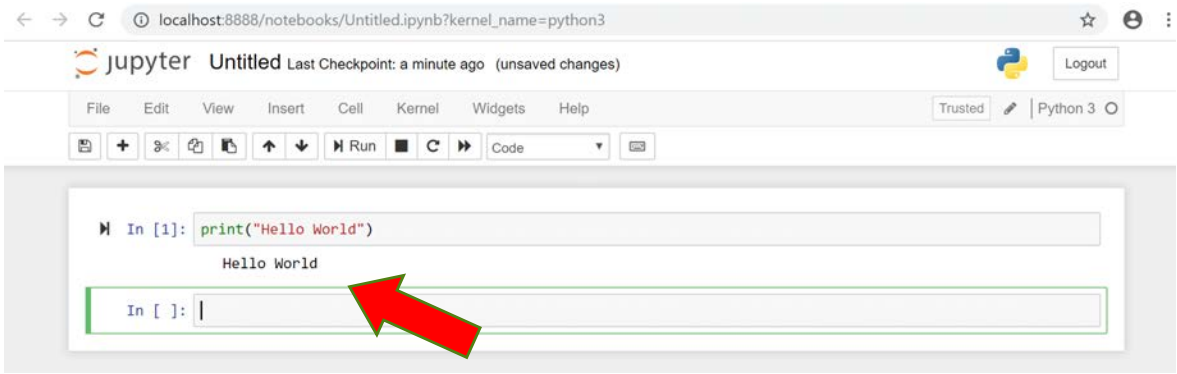
Using Jyputer Notebook

Notebook will launch with your default web browser.

Use New to start a new notebook

To run a “cell”, use shift-enter





Python For Data Science *Cheat Sheet*

Jupyter Notebook

Learn More Python for Data Science Interactively at www.DataCamp.com

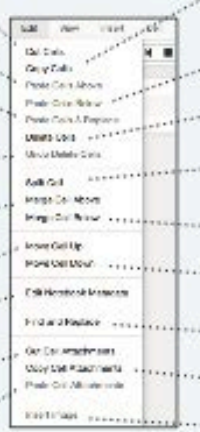
Saving/Loading Notebooks

- Create new notebook
- Make a copy of the current notebook
- Save current notebook and record checkpoint
- Preview of the printed notebook
- Close notebook & stop running any scripts
- Open an existing notebook
- Rename notebook
- Revert notebook to a previous checkpoint
- Download notebook as
  - IPython notebook
  - Python
  - HTML
  - Markdown
  - reST
  - LaTeX
  - PDF

Writing Code And Text

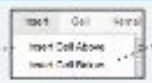
Code and text are encapsulated by 3 basic cell types: markdown cells, code cells, and raw NBConvert cells.

Edit Cells



- Cut currently selected cells to clipboard
- Paste cells from clipboard above current cell
- Paste cells from clipboard on top of current cel
- Revert "Delete Cells" invocation
- Merge current cell with the one above
- Move current cell up
- Adjust metadata underlying the current notebook
- Remove cell attachments
- Paste attachments of current cell
- Copy cells from clipboard to current cursor position
- Paste cells from clipboard below current cell
- Delete current cells
- Split up a cell from current cursor position
- Merge current cell with the one below
- Move current cell down
- Find and replace in selected cells
- Copy attachments of current cell
- Insert image in selected cells

Insert Cells



- Add new cell above the current one
- Add new cell below the current one

Working with Different Programming Languages

Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:

IPython · IRkernel · Julia

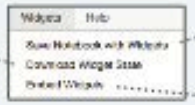
Installing Jupyter Notebook will automatically install the IPython kernel.

- Restart kernel
- Restart kernel & run all cells
- Restart kernel & run all cells
- Interrupt kernel
- Interrupt kernel & clear all output
- Connect back to a remote notebook
- Run other installed kernels

Widgets

Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc.

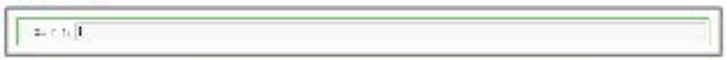
You can use them to build interactive GUIs for your notebooks or to synchronize stateful and stateless information between Python and JavaScript.



- Download serialized state of all widget models in use
- Save notebook with interactive widgets
- Embed current widgets

Command Mode:

Edit Mode:



1. Save and checkpoint
2. Insert cell below
3. Cut cell
4. Copy cell(s)
5. Paste cell(s) below
6. Move cell up
7. Move cell down
8. Run current cell
9. Interrupt kernel
10. Restart kernel
11. Display characteristics
12. Open command palette
13. Current kernel
14. Kernel status
15. Log out from notebook server

Executing Cells

- Run selected cell(s)
- Run current cells down and create a new one above
- Run all cells above the current cell
- Change the cell type of current cell
- toggle, toggle scrolling and clear all output
- Run current cells down and create a new one below
- Run all cells
- Run all cells below the current cell
- toggle, toggle scrolling and clear current outputs

View Cells

- Toggle display of Jupyter logo and filename
- Toggle line numbers in cells
- Toggle display of toolbar
- Toggle display of cell action icons:
  - None
  - Edit metadata
  - Raw cell format
  - Slideshow
  - Attachments
  - Tags

Asking For Help

- Walk through a UI tour
- Edit the built-in keyboard shortcuts
- Description of markdown available in notebook
- Python help topics
- NumPy help topics
- Matplotlib help topics
- Pandas help topics
- List of built-in keyboard shortcuts
- Notebook help topics
- Information on unofficial Jupyter Notebook extensions
- IPython help topics
- SciPy help topics
- SymPy help topics
- About Jupyter Notebook

Data Types

We shall focus on these basic data types in our workshop:

Numbers

int	for whole numbers
float	for numbers with decimal point, e.g. 5.2, 2.0

Text

str	for a sequence of characters

Containers

list	a sequence of objects, use an index to access each object

Data Types - Numbers

- Python is ‘friendly’ language
- The data type of a variable will change automatically depending on the values assigned to it.
- i is a variable
- In the 1st case, a float number is assigned to a variable i, so its data type is float.
- In the 2nd case, if an integer value is assigned to the same variable, its data type will change to integer.

```
>>> i=3.0
>>> type(i)
<class 'float'>
>>>
>>> i=3
>>> type(i)
<class 'int'>
>>>
>>> print(i/2)
1.5
>>> print(i//2)
1
>>>
```

Data Types - Strings

- Strings contain characters
- Strings can be added together with the + operator
- Strings can contain number characters, but they are not numbers (int or float)
- You can not add a number to a string
- The * operator will replicate the string with an integer value

```
>>> s = "hello"
>>> t = "world"
>>> print(s+" "+t)
hello world
>>> s = "123"
>>> print(s+4)
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
>>> print(s*4)
123123123123
>>>
```

Notes:
Strings are Unicode by default for Python 3
For string assignment, you can use “hello” or ‘hello’ but not “hello’

Data Types - List

- List is a collection of objects
- Lists can contain any type of objects
- Items in a list must be accessed by an index
- First index position starts from 0
- Python doesn't like it if you ask for something that is not in the list
- Try using a negative index, e.g. -1: What happens?

```
>>> l = [1,2,3]
>>> print(l)
[1, 2, 3]
>>> print(l[0])
1
>>> print(l[2])
3
>>> print(l[3])
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
IndexError: list index out of range
>>>
```

Data Types - List

- len gives you the length or size of list
- Get a range of items using : colon
  (This is called slicing)
  - [:3] first 3 items
  - [3:] last 3 items
- Negative index gives you items from the back
  - [-x] x^{th} last item

```
>>> l = [1,2,3,4,5,6]
>>> print(len(l))
6
>>> print(l[2:5])
[3, 4, 5]
>>> print(l[:3])
[1, 2, 3]
>>> print(l[3:])
[4, 5, 6]
>>> print(l[-1])
6
>>>
```

Range

Three versions:

- `range(y)`
  - starts at 0
  - ends before `y`
  - step up by 1
- `range(x, y)`
  - starts at `x`
  - ends before `y`
  - step up by 1
- `range(x, y, s)`
  - starts at `x`
  - ends before `y`
  - step up by `s`

```
>>> print(list(range(10)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> print(list(range(1,10)))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> print(list(range(1,10,2)))
[1, 3, 5, 7, 9]
>>>
>>> print(list(range(10,1,-1)))
[10, 9, 8, 7, 6, 5, 4, 3, 2]
>>>
```

Note: if `s` is negative, then step down by its absolute value

Data Types - Conversion

- You can convert anything into its string version:
  str(…)
- A string containing all digit characters a decimal point can be converted into a number type:
  int(…) or float(...)

```
>>> si="5"
>>> sf="5.5"
>>> print(int(si), float(sf))
5 5.5
>>> i=5
>>> print("int to str:"+str(i))
int to str:5
>>>
```

Print Formatted Numbers

Formatting numbers

%d int
%f float

Special formatting

%.2f

- float
- two digits behind the point

%+d (or f)

- force print the sign

```
>>> import math
>>> print("Pi is " + str(math.pi))
Pi is 3.141592653589793
>>> print("Pi is approx %.2f"%(math.pi))
Pi is approx 3.14
>>> print("Pos or Neg: %+d %+d"%(-5,3))
Pos or Neg: -5 +3
>>>
```

Functions

Passing parameters

- Python uses neither “pass-by-reference” nor “pass-by-value”
- It uses “pass-by-object”
- Arguments must be passed in order

```
>>> def multiply(a,b):
        return a*b

>>> multiply(2,3)
6
>>>
```

Alternatively, use parameter names to identify the arguments

```
>>> def identCar(car,colour):
...     print("Car %s has colour %s"%(car,colour))
...
>>> identCar(colour='red', car='honda')
Car honda has colour red
>>>
```

The time library

One of the functions in the time library is strftime, a flexible function to display the time based on certain format.

```
>>> print(time.strftime("Today is %d-%m-%Y %H:%M", \
... time.localtime()))
Today is 10-11-2018 15:14
>>>
```

https://docs.python.org/3/library/time.html?highlight=strftime#time.strftime

Basic Arithmetic

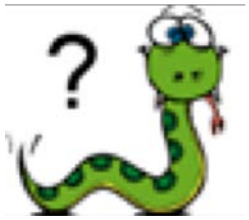
Operator Name	Code	Example When x = 2 and y = 1
Plus	x + y	x + y will give 3.
Minus	x – y	x – y will give 1.
Divide	x / y	x / y will give 2.
Multiply	x * y	x * y will give 2. You must use * instead of x for multiplication.
x to the power of y	x ** y	x ** y means 2 to power of 1 and will give 2.
Modulus	x % y	x % y will give 0. 0 is the remainder from 2 divides by 1.

https://python-reference.readthedocs.io/en/latest/docs/operators/index.html

Exercise
Homework Calculator

- Mick took 3.5 hours to finish his homework. Alice took 2.5 hours to finish her homework. Write a program to calculate the total amount of time in seconds that they took to finish their homework

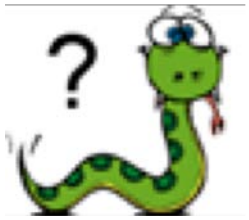
```
>>> mick = 3.5
>>> alice = 2.5
>>> total = (3.5 + 2.5) * 60 * 60
>>> print("Mick took", mick, "hr")
Mick took 3.5 hr
>>> print("Alice took", alice, "hr")
Alice took 2.5 hr
>>> print("Total time is", total, "s")
```



Exercise
Time Conversion

- Write a program (in 1 script file) to convert 1000 seconds to minutes and seconds.

```
myTime = 1000
myMins = myTime // 60
print()
mySecs = myTime % 60
print(myMins, "min &", mySecs, "sec")
```



Getting User Input

You can use input() function to ask for user input.

The value entered by the user is stored into a variable as a string.

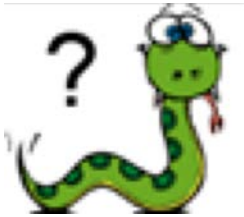
If the value is to be used as a number, you can use the int() or float() function to convert the value to the appropriate number data type.

```
>>> word = input("Enter a word:")
Enter a word:hello
>>> print(word)
hello
>>> type(word)
<class 'str'>
>>> num = input ("Enter a whole number : ")
Enter a whole number : 8
>>> print(num)
8
>>> type(num)
<class 'str'>
>>> num = int(num)
>>> print(num)
8
>>> type(num)
<class 'int'>
>>>
```

Exercise
Temperature Calculator

- The normal human body temperature is 36.9 Degree Celsius. Write a program to ask the user for name and temperature and print a message on the screen that indicate the temperature difference from the normal body temperature.

```
Enter patient's name:-John
Enter patient's temperature:-37.5
John's temperature is 0.6 degree celsius from 36.9 degree celsius.
```



If-Else Statement

https://www.youtube.com/watch?v=fVUL-vzrIcM

If-Else Statement

```
1  correct_password = "secret"
2  password = input("Enter password:-")
3
4  if password == correct_password:
5      print("You have entered correct password")
6  else:
7      print("You have entered wrong password")
```

```
Enter password:- secret
You have entered correct password
```

All lines, except line 5 are executed as **if password == correct_password** returns **True**.

```
Enter password:- letsguess
You have entered wrong password
```

All lines, except lines 6-7 are executed as **if password == correct_password** returns **False**.

Expression	What it does
a == b	Evaluates to True when a is equal to b
a != b	Evaluates to True when a is not equal to b
a < b	Evaluates to True when a is lesser than b
a > b	Evaluates to True when a is bigger than b
a <= b	Evaluates to True when a is lesser than or equal to b
a >= b	Evaluates to True when a is greater than or equal to b

If…elif…else

Type out the code below:

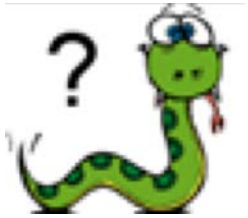
```
1   number1 = input("Enter first number:- ")
2   number2 = input("Enter second number:- ")
3   number1 = float(number1)
4   number2 = float(number2)
5
6   if number1 < number2:
7       print("First number is smaller than the second number.")
8   elif number1 > number2:
9       print("First number is greater than the second number.")
10  else:
11      print("Two numbers are equal.")
```

Exercise
BMI Calculator

Develop a BMI Calculator to calculate the BMI of a patient given the weight and height.

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)} \times \text{Height (m)}}$$

Category	Underweight	Ideal	Overweight	Obese
$\text{BMI} = \frac{weight(kg)}{height(m)^2}$	< 18	≥ 18, but < 25	≥ 25, but < 30	≥ 30



For Loops

- For loops often go hand-in-hand with lists
- Every object in the list will be processed by what is inside the for loop
- What is the data type of `i`?

Notice how each call of print at each loop will print at a different line.

How do we print numbers 0 to 9 all on the same line (0123456789)?

```
>>> numbers = range(10)
>>> for i in numbers:
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

For Loops with string

- A string is a sequence, like a list
- The for loop works similarly with strings\

```
>>> s = "freedom"
>>> for c in s:
...     print(c,end=" ")
...
f r e e d o m
>>>
```

- Slicing works for any sequence, so it works for strings too.
  - [:4] gets from the start till the fourth character
  - [-3:] gets the last third till the last character.

```
>>> s = "freedom"
>>> print(s[:4])
free
>>> print(s[-3:])
dom
>>>
```

Data Types - Dictionary

- A dictionary stores multiple key-value pairs
- E.g. In the first row of output, the dictionary contains 3 key-value pairs (which are the keys?)
- Every key is unique; no duplicate key within a dictionary
- A dictionary uses a set of curly brackets to store its key-value pairs {…}
  => Contrast with a list that uses square brackets to store its objects [...]
- To access a value in the dictionary, we use the key as an index

```
{'year': '1995', 'type_of_public_transport': 'MRT', 'average_ridership': '740000'}
{'year': '1995', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
{'year': '1995', 'type_of_public_transport': 'Bus', 'average_ridership': '3009000'}
{'year': '1995', 'type_of_public_transport': 'Taxi', 'average_ridership': '0'}
{'year': '1996', 'type_of_public_transport': 'MRT', 'average_ridership': '850000'}
{'year': '1996', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
```

Data Types - Dictionary

```
>>> scores = {'Mary': 90, 'Ben': 67, 'Jenny': 21}
>>> for s in scores:
...     print(s)
...
Mary
Ben
Jenny
```

- How does a `for` loop work on dictionaries?
- Doing '`for s in scores`' in the above code will assign the value of each key to `s`
- Change '`print(s)`' to '`print(s, scores[s])`', what do you get?

Exercise
Even Odd Counter

Write and test a program that will read 10 positive integer numbers, determine if it is even or odd, keep count of the number of even and odd numbers and display the final outcome as follows:

```
Enter number 1: 12
Enter number 2: 7
. . .
Enter number 10 : 67
Even #: 4
Odd #: 6
```

- Q: What if a user does not enter a positive integer?

Hint: use list.append() if necessary

Functions

```
>>> def myFunction():
...     print("hello")
...
>>> type(myFunction)
<class 'function'>
>>> myFunction
<function myFunction at 0x03C74738>
>>> myFunction()
hello
>>>
```

```
>>> import math
>>> def calcPHI():
...     return (math.sqrt(5)-1)/2
...
>>> calcPHI()
0.6180339887498949
```

Functions

Passing parameters

- Python uses neither “pass-by-reference” nor “pass-by-value”
- It uses “pass-by-object”
- Arguments must be passed in order

```
>>> def multiply(a,b):
        return a*b

>>> multiply(2,3)
6
>>>
```

Alternatively, use parameter names to identify the arguments

```
>>> def identCar(car,colour):
...     print("Car %s has colour %s"%(car,colour))
...
>>> identCar(colour='red', car='honda')
Car honda has colour red
>>>
```

Functions - Default Parameters

Default parameters values and checking if parameter has been passed

```
>>> def identCar(car=None,colour='red'):
...     if car == None:
...         print("You have to give me a car name")
...         return
...     print("Car %s has colour %s"%(car,colour))
...
>>> identCar(colour='blue')
You have to give me a car name
>>> identCar(car='toyota')
Car toyota has colour red
>>>
```

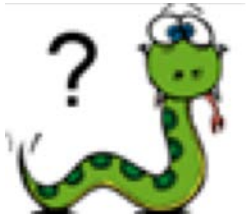
Functions - Arbitrary argument list

If you don’t know how many parameters the function will receive, you can use *args, which will be a list.

```
>>> def addAll(*args):
...     sum=0
...     for num in args:
...         sum+=num
...     return sum
...
>>> addAll(1,2)
3
>>> addAll(1,2,3,4,5,6,7,8,9)
45
>>>
```

Exercise
Arbitrary argument list

Create a function that takes in an unknown amount of parameters and returns the sum.



try .. except

- Error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks

- You can also use the finally block. The code in the finally block will be executed regardless of whether an exception occurs.

```
>>> try:
...     5/0
... except:
...     print("error")
...
error
>>>
>>> try:
...     5/0
... except Exception as e:
...     print("Exception ",type(e),": ",e.args)
...
Exception  <class 'ZeroDivisionError'> :  ('division by zero',)
>>>
```

```
>>> try:
...     5/0
... finally:
...     print("oeps, just before we run into an exception.")
...
oeps, just before we run into an exception.
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 2, in <module>
ZeroDivisionError: division by zero
>>>
```

try .. except

- A good use for try expect is to check if the user has the specific library installed and if now, explains to the user what to do:

```
>>> try:
...     import special_module
... except ImportError:
...     print("Sorry, you don't have the special_module module installed,")
...     print("and this program relies on it.")
...     print("Please install or reconfigure special_module and try again.")
...
Sorry, you don't have the special_module module installed,
and this program relies on it.
Please install or reconfigure special_module and try again.
>>>
```

- Another example is to check if a website is available:

```
1 from urllib.request import urlopen
2 def isOnline(reliableserver='http://www.google.com'):
3     try:
4         urlopen(reliableserver)
5         return True
6     except IOError:
7         return False
```

```
>>> isOnline()
True
>>>
```

String functions

- Split

```
>>> a='python or java'
>>> b=a.split(' ')
>>> type(b)
<type 'list'>
>>> b
['python', 'or', 'java']
>>>
```

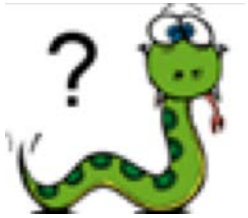
```
>>> a='python or java'
>>> b=a.split('on')
>>> b
['pyth', ' or java']
>>>
```

- Join

```
>>> a=['python','and','java']
>>> b=' '.join(a)
>>> b
'python and java'
>>> c=','.join(a)
>>> c
'python,and,java'
>>>
```

Exercise
Find Longest Word

- Create the function findLongestWord that takes in a sentence and returns the longest word. Hint: Use split()



String formatting

```
>>> import math
>>> a = math.pi
>>> a
3.141592653589793
>>> b=5
>>> c="python"
>>> line="%s %f %d"%(c,a,b)
>>> line
'python 3.141593 5'
>>>
```

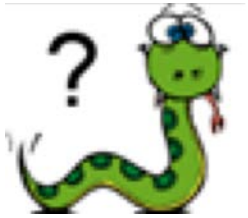
```
>>> line="%03d"%(b)
>>> line
'005'
```

Additional reading:
https://pyformat.info/
https://docs.python.org/3/library/stdtypes.html#string-formatting-operations

Exercise – Xmas Tree

- Question: Using string formatting and a loop, try to print the following xmas tree:

```
    #
   ###
  #####
 #######
#########
 ##########
```



The random library

PSA UNIVERSITY

random.randint(a, b)

Return a random integer N such that
a <= N <= b

random.random()

Return the next random floating point number in the range [0.0, 1.0]

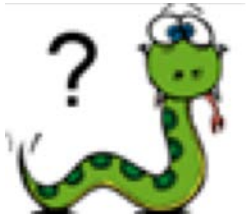
Other random functions

random.shuffle(List)
random.choice(List)

More at http://docs.python.org/library/random.html

Exercise - Guessing Game

- Create a random number between 1 and 20 and prompt the user to guess the secret number. He is allowed a maximum of 6 guesses after which the secret number will be displayed and the program exits. For every guess, the program will display a message saying if the number guessed is higher or lower than the secret number. If he guessed the correct number, the program will display the number of tries he had taken and the program exits.



Exercise - Guessing Game

- Sample output

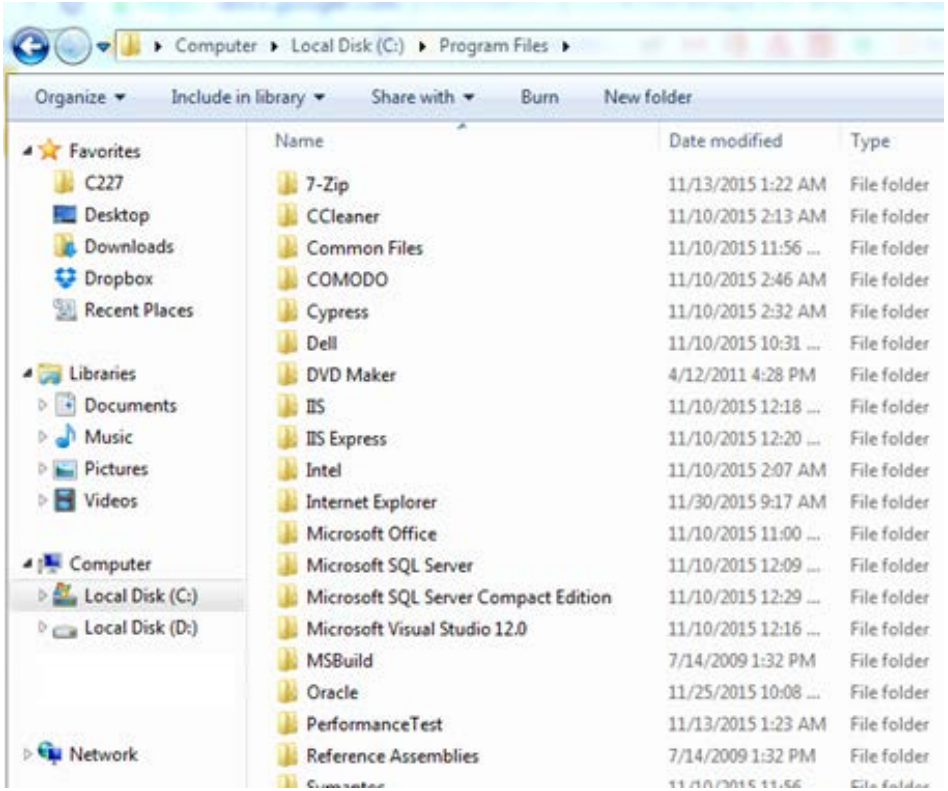
```
What is your name?
John
Well, John, I am thinking of a number between 1 and 20
Take a guess
5
Your guess is too low.
Take a guess
10
Your guess is too low.
Take a guess
15
Your guess is too high.
Take a guess
12
Your guess is too low.
Take a guess
14
Good job, John! You guessed my number in 5 guesses!

Process finished with exit code 0
```

```
What is your name?
John
Well, John, I am thinking of a number between 1 and 20
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
nope. The number I was thinking of was 6

Process finished with exit code 0
```

Search for file by filename

Let’s say you want to find a file by filename “readme.txt” and it’s somewhere in

C:\Program Files\



Search for file by filename

We need to work with the os library, so we import it at the start.

The folder we search in is stored in the variable where.

At line 6 we create a new function called searchByName.
It has one parameter, which is name and contains the name of the file we are looking for.

In this function we walk through all directories and files.
Then for each file we compare if it is the filename we are looking for.

```
1   import os
2
3   where = "C:\\Program Files\\"
4   #where = "C:\\Program Files (x86)\\"
5
6   def searchByName(name):
7       for root, dirs, files in os.walk(where):
8           for file in files:
9               if file == name:
10                  print (os.path.join(root, file))
11
12  searchByName("readme.txt")
13
```

Search for file by filename

Let's say you want to know the combined filesize of all these readme.txt files.

Initialize a new variable totalSize to 0 at the start of our function.

Then for each readme.txt we add the size to this variable with
totalSize += os.path.getsize(os.path.join(root,file))

Then we make this function return this value with return totalSize at the very end of the function with
return totalSize

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                print(os.path.join(root,file))
                totalSize += os.path.getsize(os.path.join(root,file))
    return totalSize
```

Make the amendments as indicated above, then run it and observe.

Did it show you the total ?

Search for file by filename

The function returned the value, but we didn't do anything with it.

When we call the function, we can assign the value to a variable. Then we can print the content.

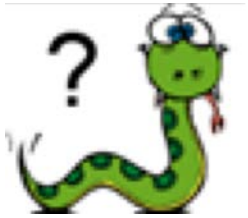
```
total = searchByName("readme.txt")
print ("Total is : %d" % (total))
print ("All done")
```

```
C:\Program Files\Unity\MonoDevelop\Addins\MonoDevelop.AspNet\Schemas\readme.txt
C:\Program Files\Unity\MonoDevelop\Addins\MonoDevelop.XmlEditor\schemas\readme.txt
Total is 17569
All done.
>>>
```

Exercise – listing files

Can you update the program to show the individual file size of all the files?

```
>>>
  1761 C:\Program Files\7-Zip\readme.txt
    83 C:\Program Files\Unity\Editor\Data\Playba
    62 C:\Program Files\Unity\Editor\Data\Playba
   549 C:\Program Files\Unity\Editor\Data\Playba
   695 C:\Program Files\Unity\Editor\Data\Playba
   126 C:\Program Files\Unity\Editor\Data\Playba
    89 C:\Program Files\Unity\Editor\Data\Playba
    66 C:\Program Files\Unity\Editor\Data\Playba
   250 C:\Program Files\Unity\Editor\Data\Playba
    25 C:\Program Files\Unity\Editor\Data\Playba
 11333 C:\Program Files\Unity\Editor\Data\Playba
   718 C:\Program Files\Unity\Editor\Data\Playba
   906 C:\Program Files\Unity\MonoDevelop\Addins
   906 C:\Program Files\Unity\MonoDevelop\Addins
Total is 17569
All done.
>>>
```



Search for file by filename

Since we use the full filename multiple times, it makes sense to store it in a separate variable. Same for the filesize.

The %6d makes sure the output looks nice, in columns format.

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                fullName = os.path.join(root,file)
                fileSize = os.path.getsize(fullName)
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

File files larger than xxMB

Create a new function called searchBySize, that takes one parameter and only prints those files larger than that parameter.

for example:
searchBySize(50000000)
will only print the full path and filename of those files that exceed 50Mb

You still need the if statement but it's not based on the name.
Do you know the fileSize at the point of the if statement ?

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                fullName = os.path.join(root,file)
                fileSize = os.path.getsize(fullName)
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

File files larger than xxMB

Most of the function can be copied, but you need to move the declaration of fullName and fileSize before the if statement.

The if statement then can use the fileSize.

You can still return the totalSize although that was not required.

```
def searchBySize(size):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if fileSize > size:
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

File files of certain file type

What if you want to find all the files of specific file type ?

Create another function (copy the last) and call it searchByExtension
It has to take one parameter which is the extension to look for.

We can use the buildin function .endswith(“.doc”) in our if statement to compare:
if file.endswith(“.doc”):

You should be able to call this function like:
searchByExtension(“.doc”)

```
def searchByExtension(ext):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext):
                    print("%6d %s"%(fileSize, fullName))
                    totalSize += fileSize
    return totalSize
```

Again the core is the same, but the if statement is different.

There are other ways to solve this, but this is quite straight forward

Find in files of certain file type

Copy the last function and call it searchByContent.

This new function has to take one more parameter (call it keyword) which is the string.
This string has to be present in the file in order to be counted.

You can combine two conditions in an if statement with and :

if …. and …. :
print “okay”

```
def searchByContent(ext, keyword):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext) and keyword in open(fullName, encoding="Latin-1").read():
                    print("%6d %s"%(fileSize, fullName))
                    totalSize += fileSize
    return totalSize
```

You should be able to use it like this:
searchByContent(".txt","Copyright")

To search in text files you can use
if keyword in open(fullName).read():

Find in files of certain file type

```
def searchByContent(ext, keyword):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext) and keyword in open(fullName, encoding="Latin-1").read():
                    print("%6d %s"%(fileSize, fullName))
                    totalSize += fileSize
    return totalSize
```

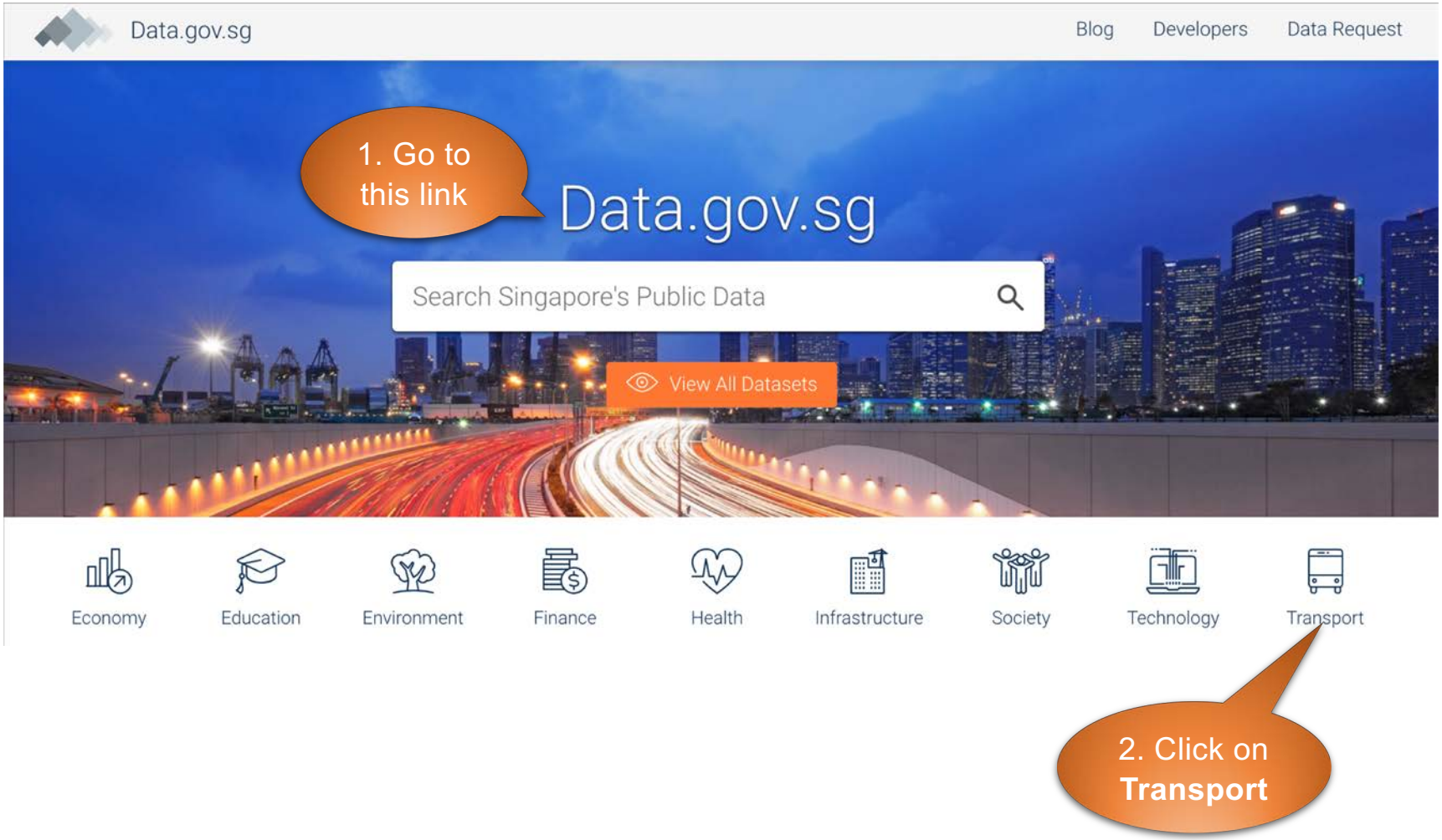
And this is how it looks like.
This function is slower because it has to go through all the content in the files.

open(fullName).read() literally represents the whole file!
Large files might be a problem because they can not fit in memory at one time.

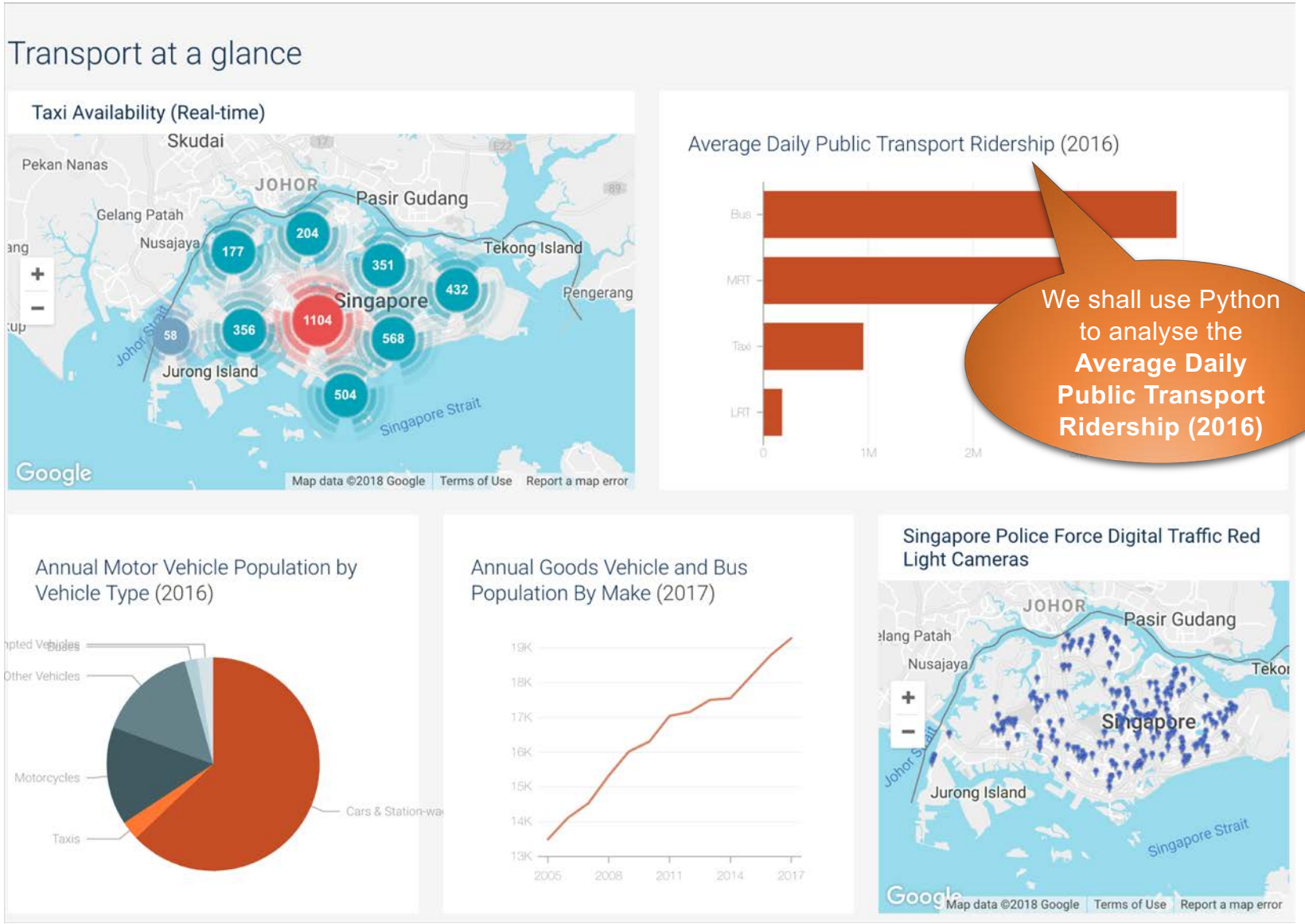
A better solution would be to load blocks of content, but we save that for another course.

Note:
In your if statement, put the file.endswith(ext) first.
This way it will not execute the second part if the first part is already false and thus save precious time.

Use Case


Data.gov.sg
Blog
Developers
Data Request
1. Go to this link
Data.gov.sg
Search Singapore's Public Data
View All Datasets
Economy
Education
Environment
Finance
Health
Infrastructure
Society
Technology
Transport
2. Click on **Transport**

Use Case

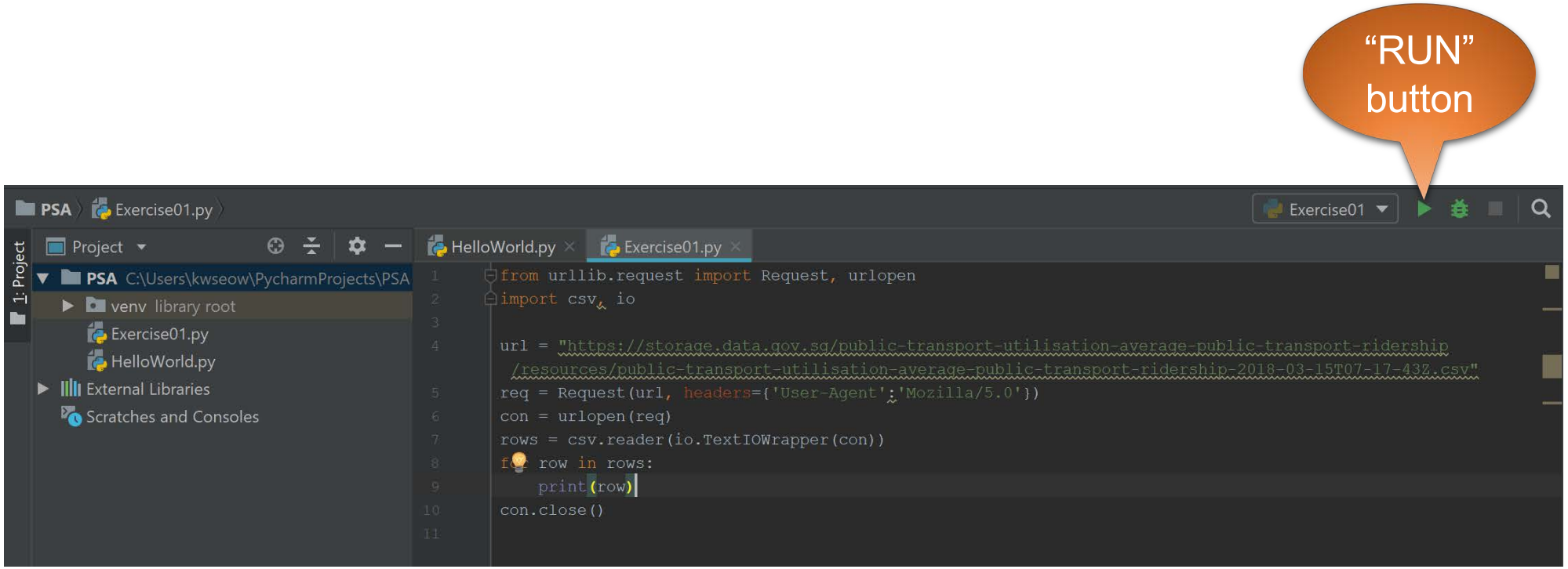

Transport at a glance
Taxi Availability (Real-time)
Skudai
Pekan Nanas
JOHOR
Pasir Gudang
Gelang Patah
Nusajaya
Tekong Island
Pengerang
Singapore
Jurong Island
Singapore Strait
Johor Strait
204
177
351
432
1104
356
58
568
504
Google
Map data ©2018 Google
Terms of Use
Report a map error
Average Daily Public Transport Ridership (2016)
Bus
MRT
Taxi
LRT
0
1M
2M
We shall use Python to analyse the Average Daily Public Transport Ridership (2016)
Annual Motor Vehicle Population by Vehicle Type (2016)
Other Vehicles
Motorcycles
Taxis
Annual Goods Vehicle and Bus Population By Make (2017)
19K
18K
17K
16K
15K
14K
13K
2005
2008
2011
2014
2017
Singapore Police Force Digital Traffic Red Light Cameras
JOHOR
Pasir Gudang
Nusajaya
Singapore
Jurong Island
Singapore Strait
Google
Map data ©2018 Google
Terms of Use
Report a map error

Use Case

```
  - ''
  - '* Only data from 2001 onwards had been updated when the methodology of estimating
    taxi ridership was revised in 2003.'
  - ''
Topics:
  - 'Transport'
Keywords:
  - 'Public Transport Utilisation'
Publisher:
  Name: 'Land Transport Authority'
  Admin 1:
    Name: 'Land Transport Authority'
    Department: 'Land Transport Authority'
    Email: 'Datamall@LTA.gov.sg'
Sources:
  - 'Land Transport Authority'
Source Url: 'http://www.mytransport.sg/content/mytransport/home/dataMall.html#Facts_&_Figures'
License: 'https://data.gov.sg/open-data-licence'
Frequency: 'Annual'
Coverage: '1995-01-01 to 2016-12-31'
Last Updated: '2018-03-15T07:17:43.880477'
Resources:
  -
    Identifier: '552b8662-3cbc-48c0-9fbb-abdc07fb377a'
    Title: 'Public Transport Utilisation - Average Public Transport Ridership'
    Url: 'https://storage.data.gov.sg/public-transport-utilisation-average-public-transport-ridership/resources/
public-transport-utilisation-average-public-transport-ridership-2018-03-15T07-17-43Z.csv'
    Format: 'CSV'
    Coverage: '1995-01-01 to 2016-12-31'
    Last Updated: '2018-03-15T07:17:43.352991'
    Schema:
      -
        Name: 'year'
        Title: 'Year'
        Type: 'datetime'
        Sub Type: 'year'
        Format: 'YYYY'
      -
        Name: 'type_of_public_transport'
```

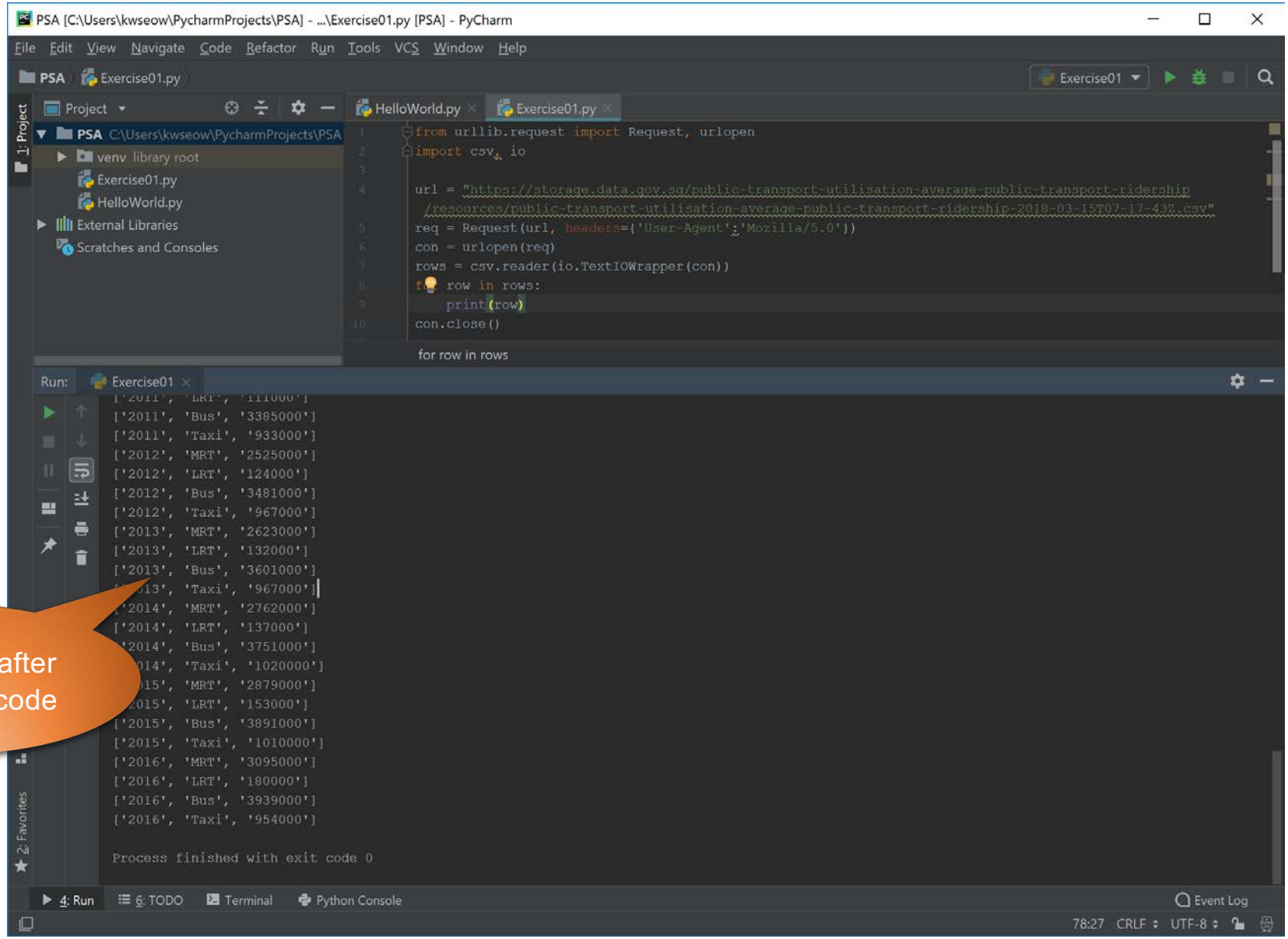
Copy this Url link

Use Case



Type this code into Pycharm Editor and click the **Run** button
(Replace the string in `url` variable with what you had copied from the txt file)

Use Case


PSA [C:\Users\kwseow\PycharmProjects\PSA] - ...\Exercise01.py [PSA] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
PSA Exercise01.py
Exercise01
Project
HelloWorld.py
Exercise01.py
PSA C:\Users\kwseow\PycharmProjects\PSA
venv library root
Exercise01.py
HelloWorld.py
External Libraries
Scratches and Consoles
from urllib.request import Request, urlopen
import csv, io
url = "https://storage.data.gov.sg/public-transport-utilisation-average-public-transport-ridership
/resources/public-transport-utilisation-average-public-transport-ridership-2018-03-15T07-17-43Z.csv"
req = Request(url, headers={'User-Agent':'Mozilla/5.0'})
con = urlopen(req)
rows = csv.reader(io.TextIOWrapper(con))
for row in rows:
print(row)
con.close()
for row in rows
Run: Exercise01
['2011', 'Bus', '3385000']
['2011', 'Taxi', '933000']
['2012', 'MRT', '2525000']
['2012', 'LRT', '124000']
['2012', 'Bus', '3481000']
['2012', 'Taxi', '967000']
['2013', 'MRT', '2623000']
['2013', 'LRT', '132000']
['2013', 'Bus', '3601000']
['2013', 'Taxi', '967000']
['2014', 'MRT', '2762000']
['2014', 'LRT', '137000']
['2014', 'Bus', '3751000']
['2014', 'Taxi', '1020000']
['2015', 'MRT', '2879000']
['2015', 'LRT', '153000']
['2015', 'Bus', '3891000']
['2015', 'Taxi', '1010000']
['2016', 'MRT', '3095000']
['2016', 'LRT', '180000']
['2016', 'Bus', '3939000']
['2016', 'Taxi', '954000']
Process finished with exit code 0
4: Run
6: TODO
Terminal
Python Console
Event Log
78:27 CRLF UTF-8
Your output after running the code

Use Case

```
1   from urllib.request import Request, urlopen
2   import csv, io
3
4   url = "https://storage.data.gov.sg/public-transport-
     /resources/public-transport-utilisation-average-pub
5   req = Request(url, headers={'User-Agent':'Mozilla/5.
6   con = urlopen(req)
7   rows = csv.reader(io.TextIOWrapper(con))
8   for row in rows:
9       print(row)
10  con.close()
```

What is Happening Here?

Line	Explanation
1	Load url modules
2	Load csv and io modules
4	Store web address in `url`
5	Add header to our 'browser'
6	Open and save the connection in `con`
7	Save the output in list variable `rows`
8	Go through all rows
9	Print each row
10	Close the connection

Quick question: What is the data type of `row`?

Use Case

- We want to show only data after 2010

Current code:

```
1   from urllib.request import Request, urlopen
2   import csv, io
3
4   url = 'https://storage.data.gov.sg/public-transport-utilisation-avera
5   req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6   con = urlopen(req)
7   rows = csv.reader(io.TextIOWrapper(con))
8   for row in rows:
9       print(row)
10  con.close()
```

Something is needed to decide when to print

Recall the data type of row: Remember what it is?

Use Case

```
1  from urllib.request import Request, urlopen
2  import csv, io
3
4  url = "https://storage.data.gov.sg/public-transport-utilisation-average-pu
5  req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6  con = urlopen(req)
7  rows = csv.reader(io.TextIOWrapper(con))
8  for row in rows:
9      target_year = int(row[0])
10     if target_year > 2010:
11         print(row)
12
13 con.close()
```

Add these lines in the for loop, and indent the print statement within it

- `row` is a list type
- The year is stored as the **first** object in the list, so use index 0
- Recall the year value in the output has quotes around it, e.g. '1995'
- Need to convert it to int to compare with the number 2010
- So we will only print `row` if the condition is fulfilled

Now run the program…

Use Case

Uh oh! What went wrong? :(

```
C:\Python36\python.exe C:/Users/jason_lim/PycharmProjects/MyFirstProject/exercise1.py
Traceback (most recent call last):
  File "C:/Users/jason_lim/PycharmProjects/MyFirstProject/exercise1.py", line 9, in <module>
    target_year = int(row[0])
ValueError: invalid literal for int() with base 10: 'year'

Process finished with exit code 1
```

Check the first line in your output!


```
['year', 'type_of_public_transport', 'average_ridership']
['1995', 'MRT', '740000']
['1995', 'LRT', '0']
['1995', 'Bus', '3009000']
['1995', 'Taxi', '0']
```

Attempting to convert a string with non-digits into an integer will cause an error…

Use Case

```
1  from urllib.request import Request, urlopen
2  import csv, io
3
4  url = 'https://storage.data.gov.sg/public-transport-utilis
5  req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6  con = urlopen(req)
7  rows = csv.DictReader(io.TextIOWrapper(con))
8  for row in rows:
9      print(dict(row))
10 con.close()
```

1. `csv` module has a method called `DictReader`

2. Convert `row` into `dict` type

- Use `DictReader` method from `csv` module to convert each row into a dictionary type
- This is useful when the CSV file contains a header
- So, what is a dictionary and how does it differ from a list?

Output

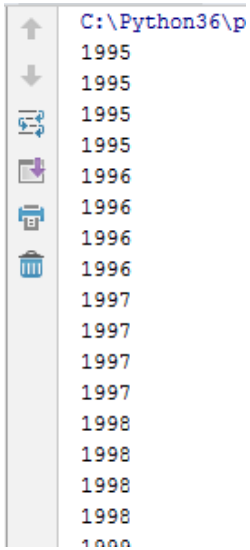
```
{'year': '1995', 'type_of_public_transport': 'MRT', 'average_ridership': '740000'}
{'year': '1995', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
{'year': '1995', 'type_of_public_transport': 'Bus', 'average_ridership': '3009000'}
{'year': '1995', 'type_of_public_transport': 'Taxi', 'average_ridership': '0'}
{'year': '1996', 'type_of_public_transport': 'MRT', 'average_ridership': '850000'}
{'year': '1996', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
```

Use Case

```
1  from urllib.request import Request, urlopen
2  import csv, io
3
4  url = "https://storage.data.gov.sg/public-transport-utilisation-ave
5  req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6  con = urlopen(req)
7  rows = csv.DictReader(io.TextIOWrapper(con))
8  for row in rows:
9      print(row['year'])
10 con.close()
11
```

This will print only the years

Next we will add an if statement to find years>2010



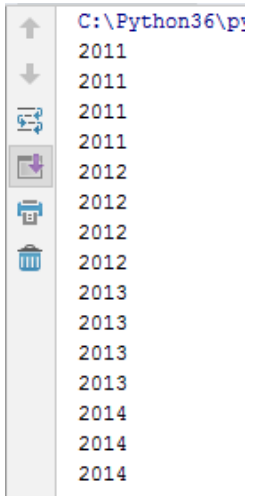
Use Case
Now Insert the Condition

```
1  from urllib.request import Request, urlopen
2  import csv, io
3
4  url = "https://storage.data.gov.sg/public-transport-utilisation-aver
5  req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6  con = urlopen(req)
7  rows = csv.DictReader(io.TextIOWrapper(con))
8  for row in rows:
9      if int(row['year']) > 2010:
10         print(row['year'])
11 con.close()
12
```

1. Add condition

2. This print statement will only execute if it satisfies the condition

- The key to get the value of year is the string `'year'`
- Remember the year values are stored as strings, so we need to convert them into integers

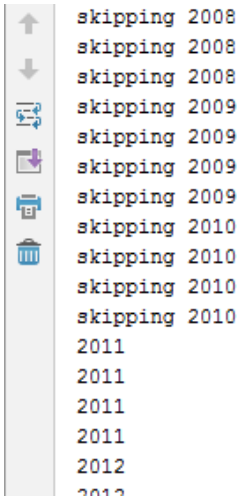


Use Case
If-Else Modified

```
1  from urllib.request import Request, urlopen
2  import csv, io
3
4  url = "https://storage.data.gov.sg/public-transport-utilisation-aver
5  req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
6  con = urlopen(req)
7  rows = csv.DictReader(io.TextIOWrapper(con))
8  for row in rows:
9      if int(row['year']) > 2010:
10         print(row['year'])
11     else:
12         print ("skipping", row['year'])
13 con.close()
```

All rows with years that are <= 2010 will end up here

- Suppose we would like to display a message for the rows we skipped
- We can specify a Else branching



Use Case Enhancement

Hands-on (You're on your own!):

Change your program such that it will only show the year and average ridership for MRT

Note: Use == to compare for equal values in Python

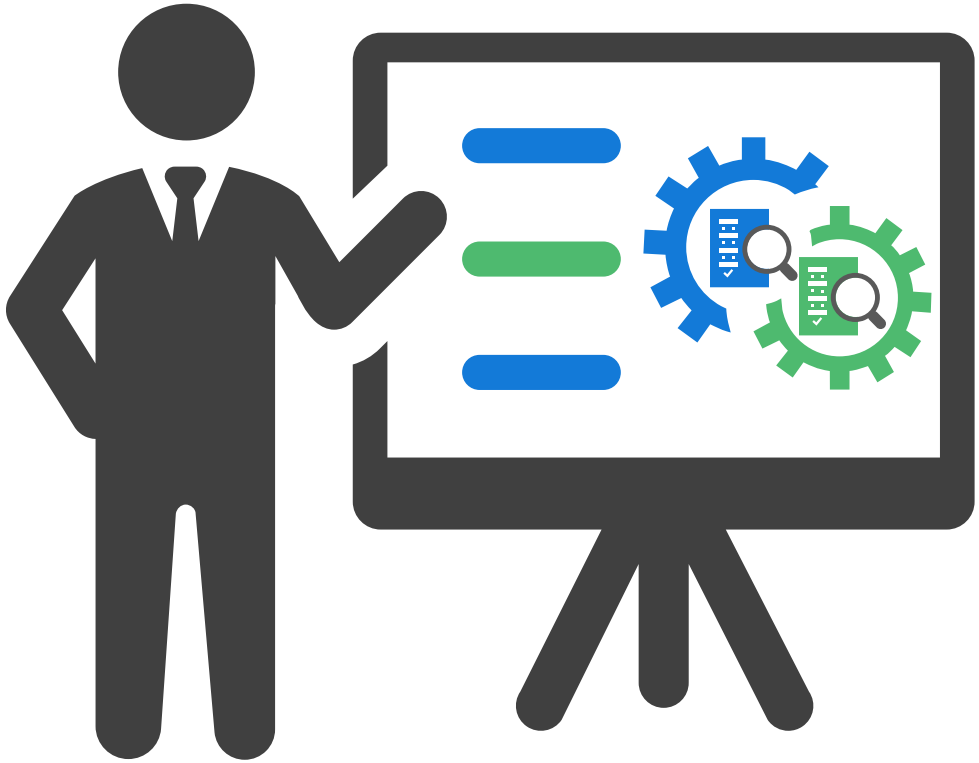
```
1995 740000
1996 850000
1997 911000
1998 946000
1999 986000
2000 1047000
2001 1071000
2002 1081000
2003 1171000
2004 1270000
2005 1321000
2006 1408000
2007 1527000
2008 1698000
2009 1782000
2010 2069000
2011 2295000
2012 2525000
2013 2623000
2014 2762000
2015 2879000
2016 3095000

Process finished with exit code 0
```

Summary

Think Python is an introduction to Python programming for beginners. It starts with basic concepts of programming, and is carefully designed to define all terms when they are first used and to develop each new concept in a logical progression. Larger pieces, like recursion and object-oriented programming are divided into a sequence of smaller steps and introduced over the course of several chapters.

PSA UNIVERSITY

Day 1 Summary



- ✓ *Basics on Python*
- ✓ *Development Environment*
- ✓ *Datatypes (basic, list, dictionary)*
- ✓ *Printing*

- ✓ *Functions*
- ✓ *Time library*
- ✓ *Basic Arithmetic*
- ✓ *Getting user inputs*

- ✓ *If-Else statement*
- ✓ *For Loops*
- ✓ *Exception handling*
- ✓ *File management*

- ✓ *[Optional] Analytic Use case*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: http://bit.ly/2zFvbsi


home
work

Thank you