PSA UNIVERSITY

REPUBLIC POLYTECHNIC
DISCOVER. TRANSFORM. ACHIEVE

Introductory Programming in Python

Day 1

python™

Quick Survey

http://bit.ly/2Y8MTi2

Introduction of trainer

Name
Seow Khee Wei

Email
seow_khee_wei@rp.edu.sg

Phone
98463112

Telegram
@kwseow

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!

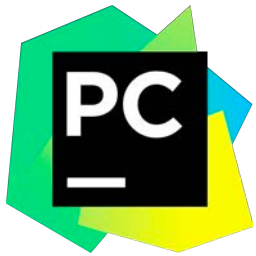
Approach

- 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:

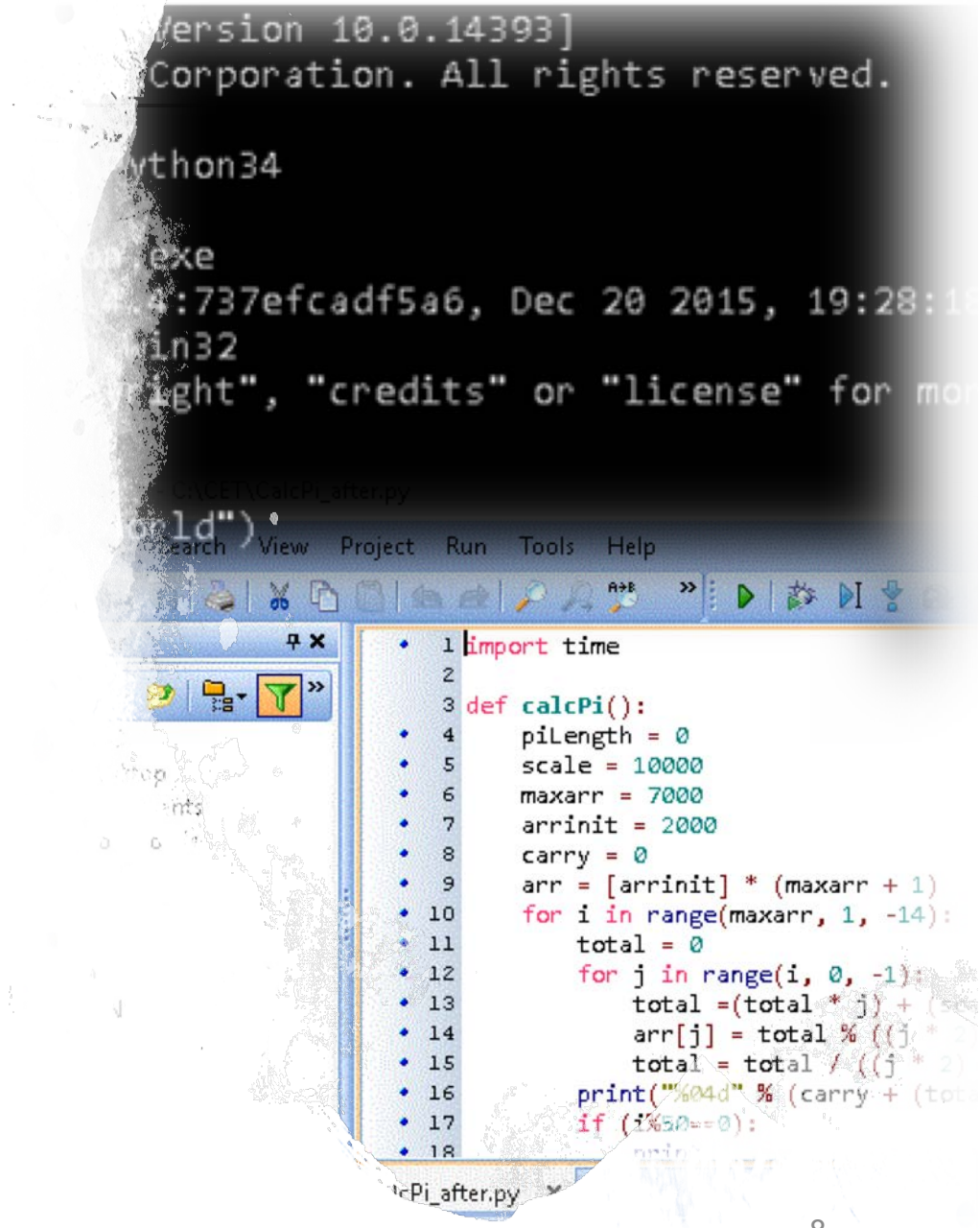
- Your laptop works



- 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

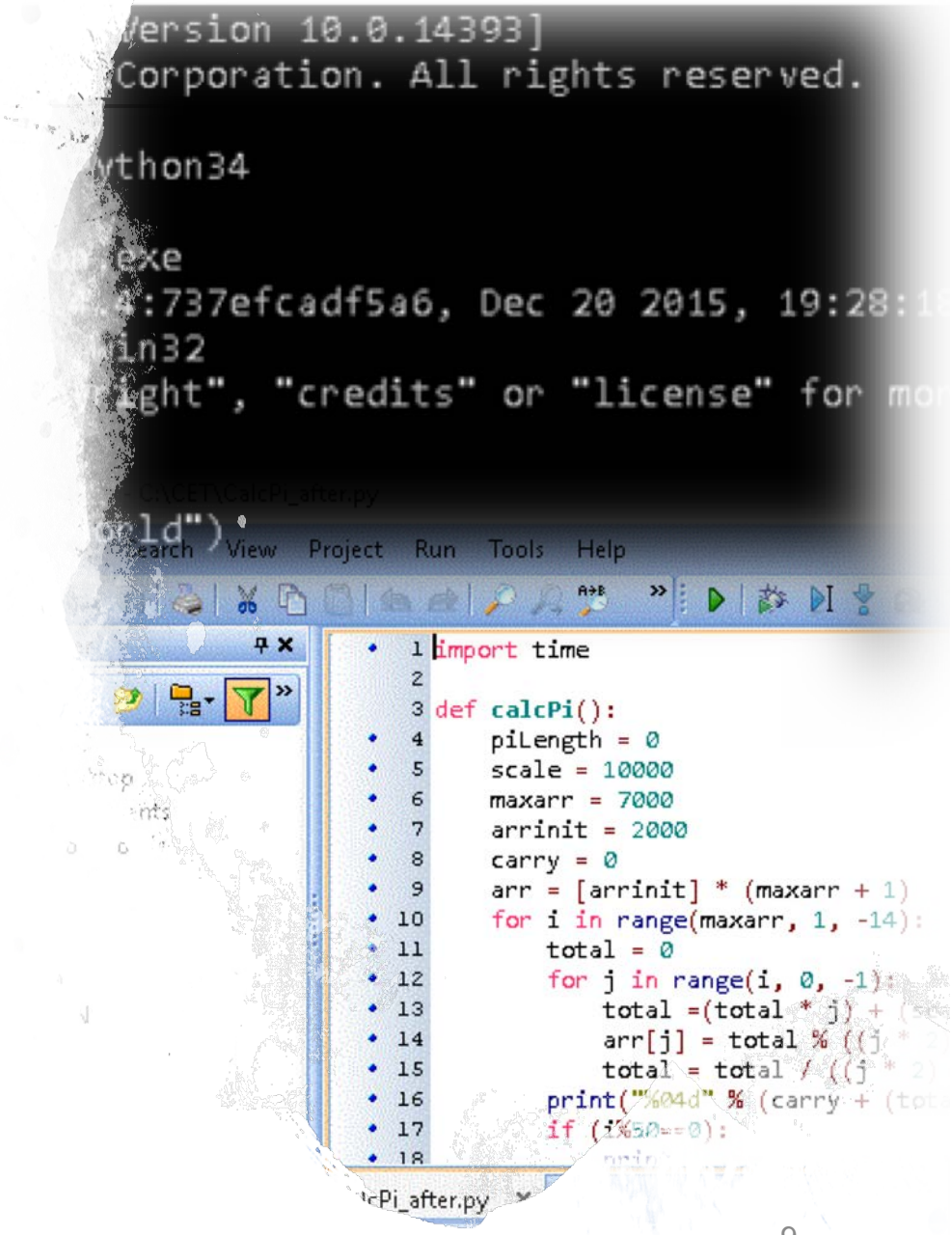
Programme Day One

- A brief history of Python
- Setting up python environment
- Learn the basics:
  - Data types
  - Conversions
  - String operations
  - Functions
  - And more!
- Try/except
- String functions, formatting
- Graphical User Interface



Programme
Day Two

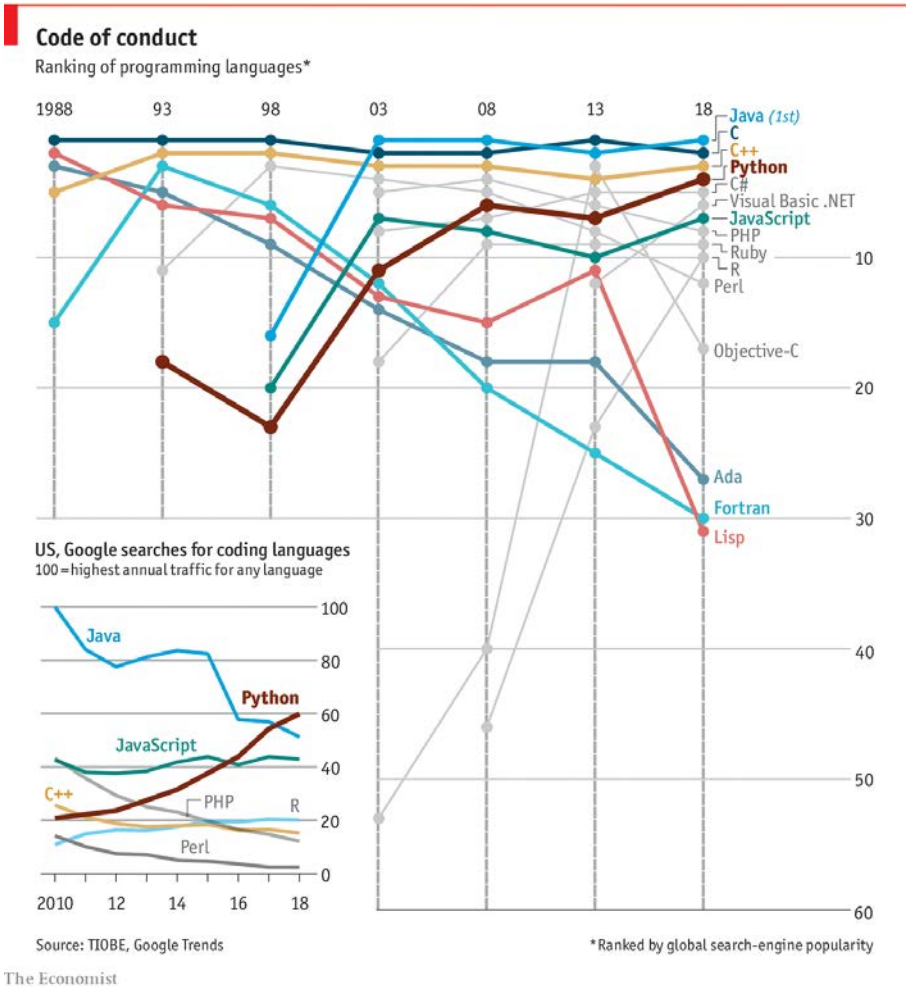
- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Processing CSV files
- Image processing: loading, scaling, watermark, applying filters
- Connecting to the Web
- Sending emails
- Telegram bot



Introduction to Python

What is Python?

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

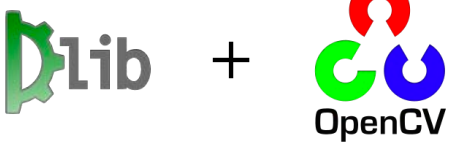


Introduction to Python

- 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)

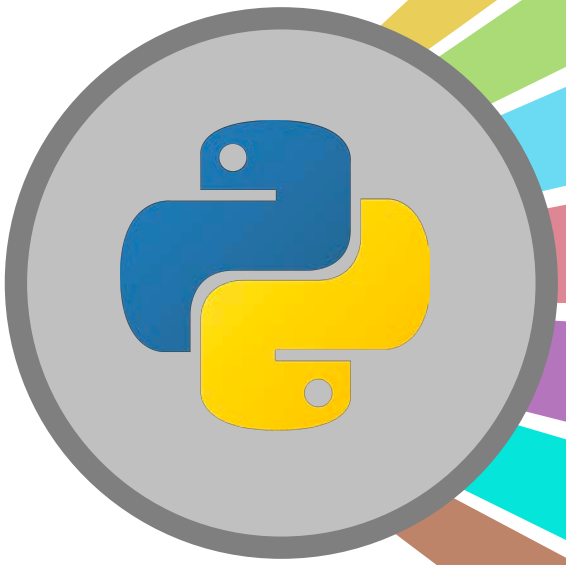
06 ***Education.*** University of California, Irvine. University of New South Wales (Australia), NUS, SUTD, SMU, RP

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

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

Introduction to Python

What is Python used for?



01 | Web and internet development (Django, Flask, Google App Engine)

02 | File storage (Dropbox)

03 | Scientific and Numeric computing (SciPy, NumPy)

04 | Data Science, Artificial Intelligence, Machine Learning (Keras, Scikit-learn, Pandas)

05 | Micro Controller (Micro-Python)

06 | Embedded in software packages (Maya, Blender)

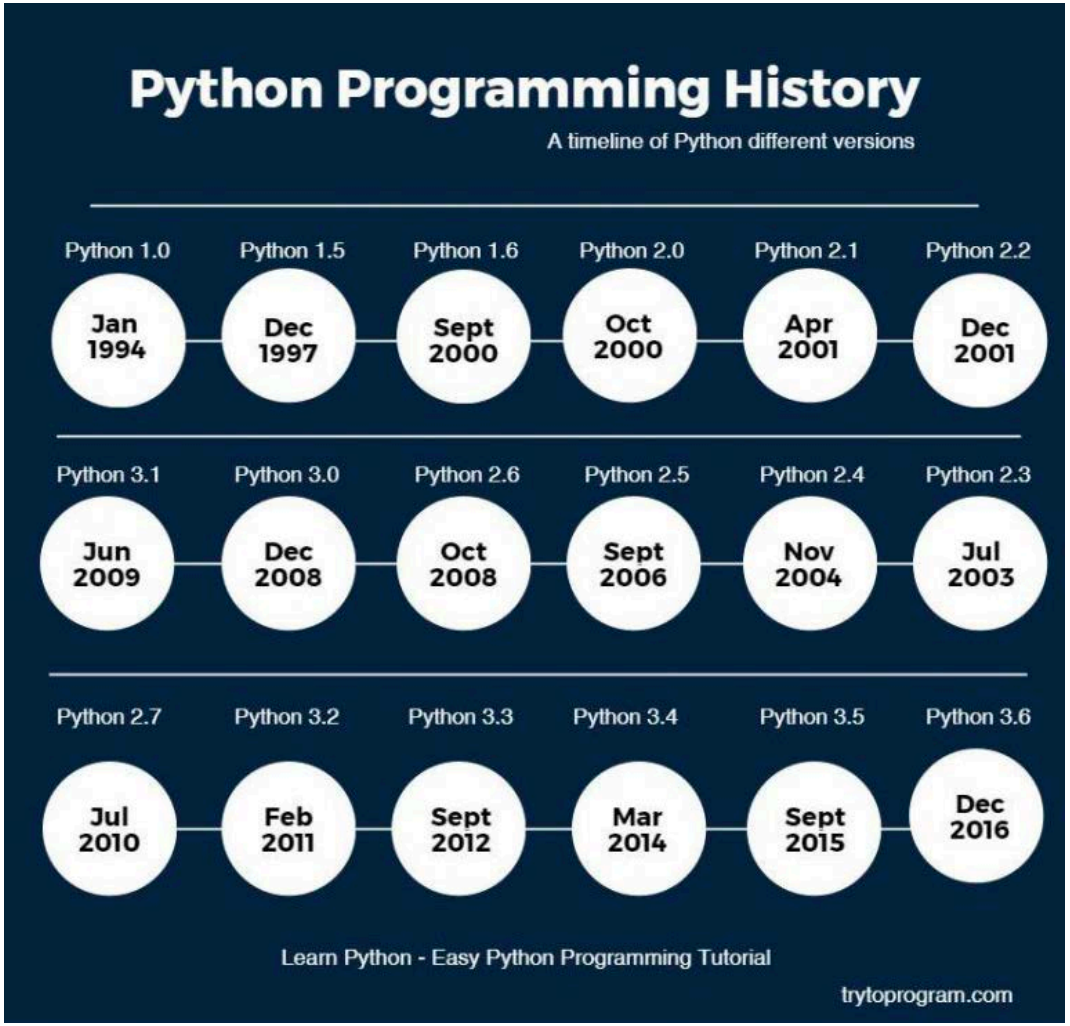
07 | Scripting to perform simple (but mundane & repetitive) tasks

Introduction to Python

Why the name, Python?

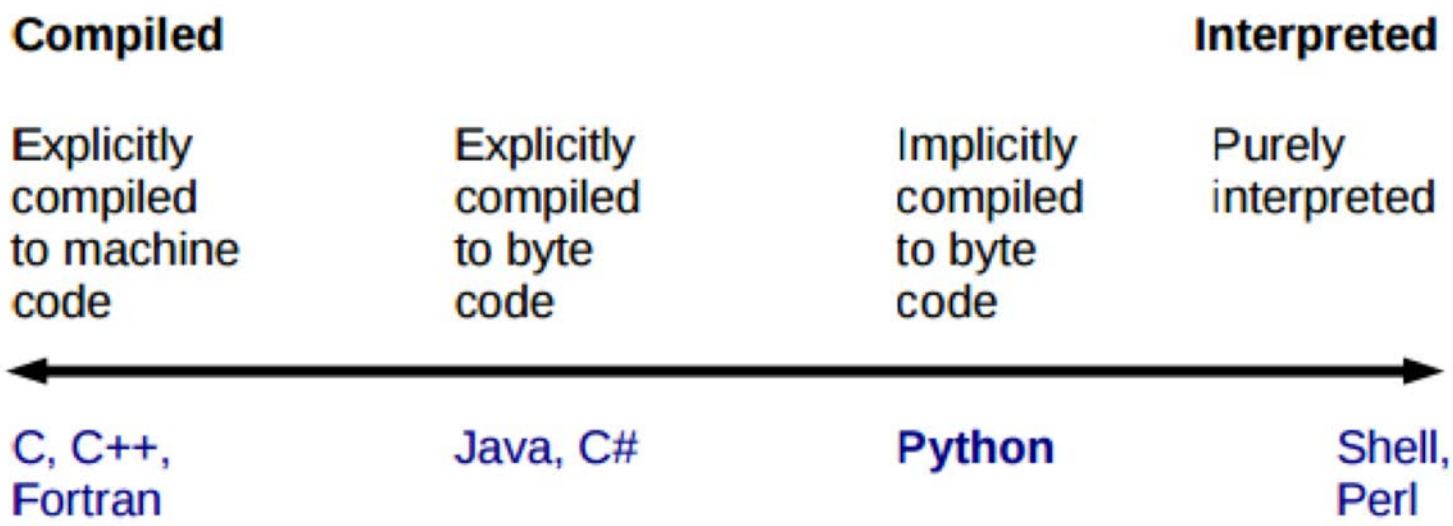
- Originally not a snake, but from the British comedy "**Monty Python's Flying Circus**". The snake logo came later.
- Invented in 1990 by Guido Van Rossum
- Initially intended to be a scripting language on Amoeba OS
- Python was influenced by ABC and Modula-3
- First public release was in 1991

Introduction to Python



Python has two versions currently: **2.7.14 and 3.8**

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**

Why Python

- Focus on problem solving, and not on programming syntax

```python
width = input("Enter Width: ")
height = input("Enter Height: ")

area = float(width) * float(height)
print("Area: " + str(area))
```

```java
import java.util.Scanner;

public class AreaApp {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter Width: ");
        double width = scanner.nextDouble();

        System.out.println("Enter Height: ");
        double height = scanner.nextDouble();

        double area = width * height;

        System.out.println("Area: " + area);
    }
}
```

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

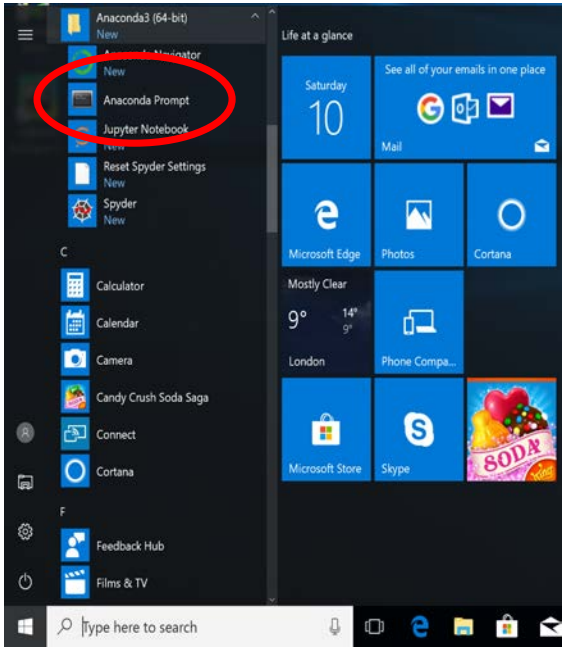
- Use a text editor (notepad, vi, emacs, etc)
- Use an IDE (IDLE, PyScripter, PyCharm, etc)

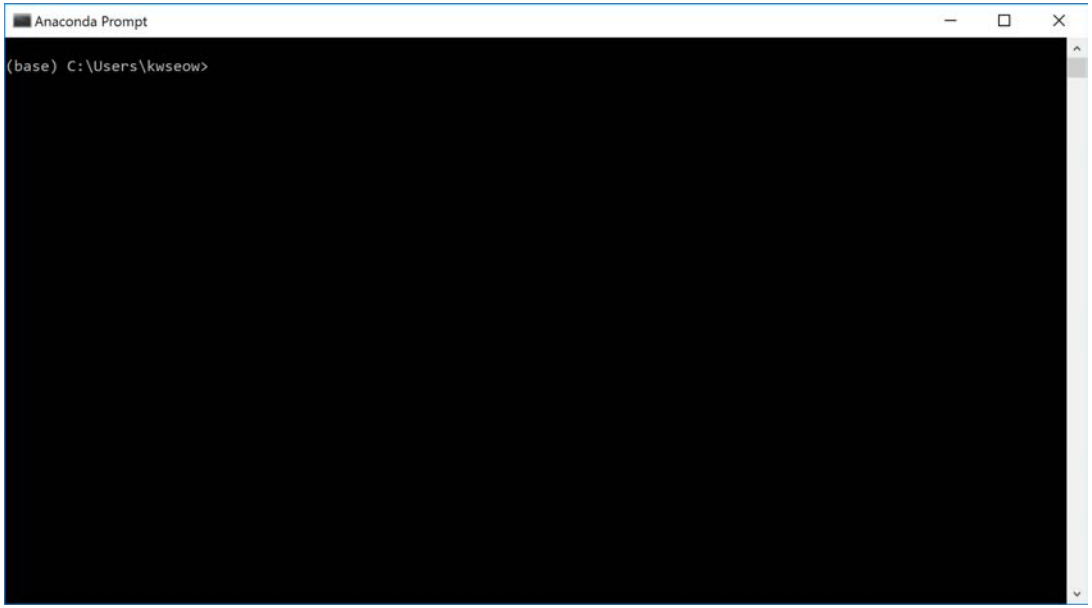
Experiment with Jupyter Notebook

- Installed together with Anaconda

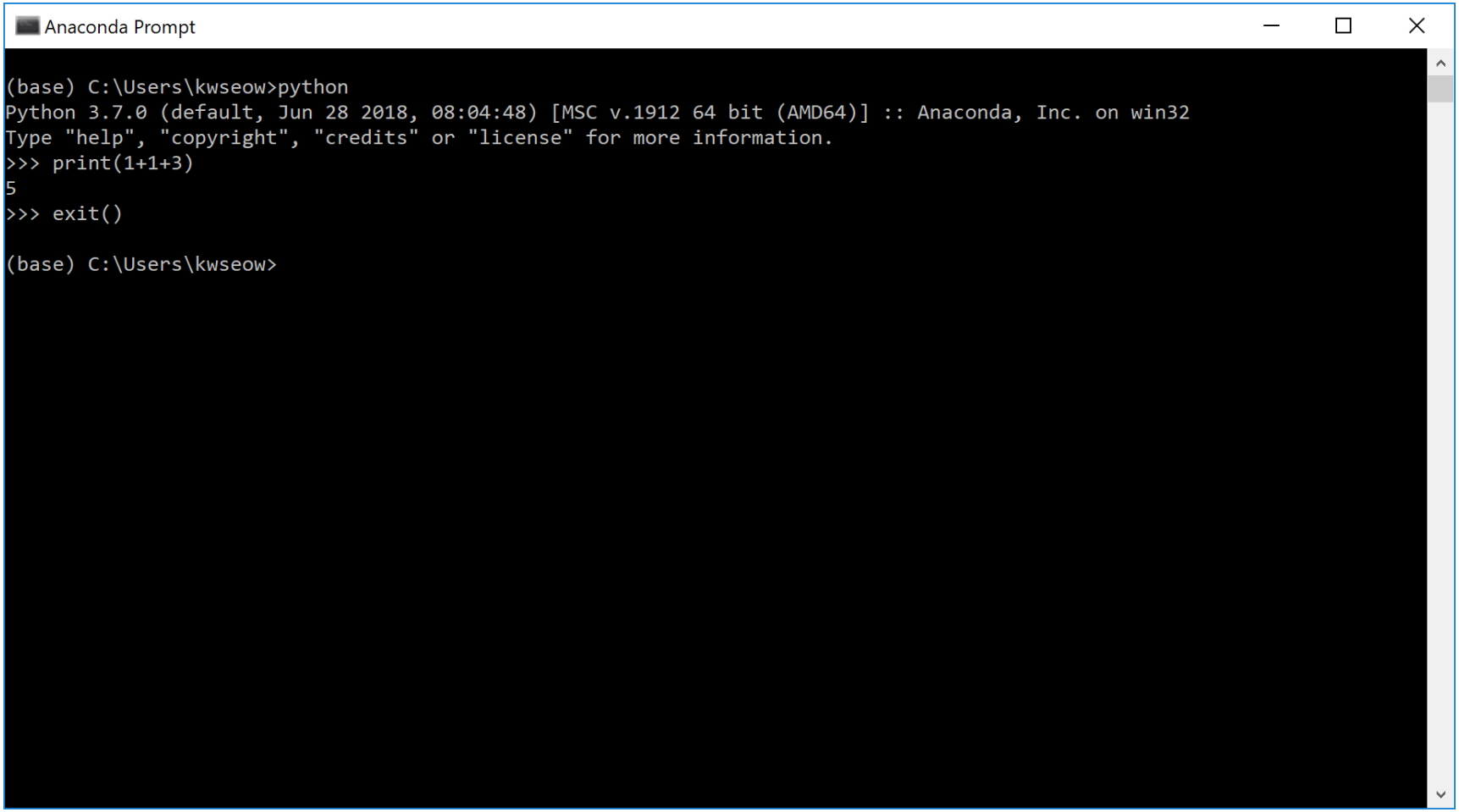
Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed 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

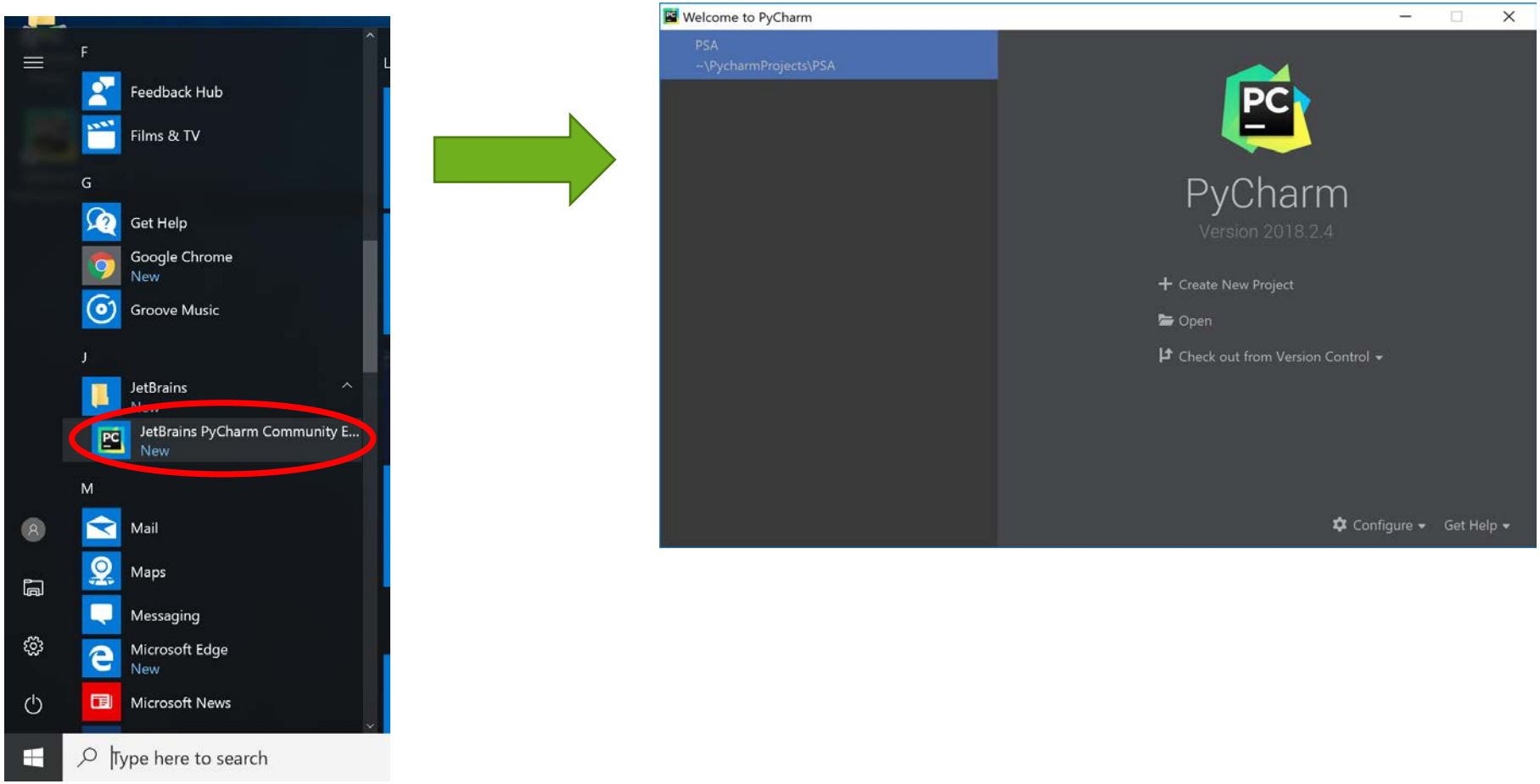




First, Experience the Basic Way



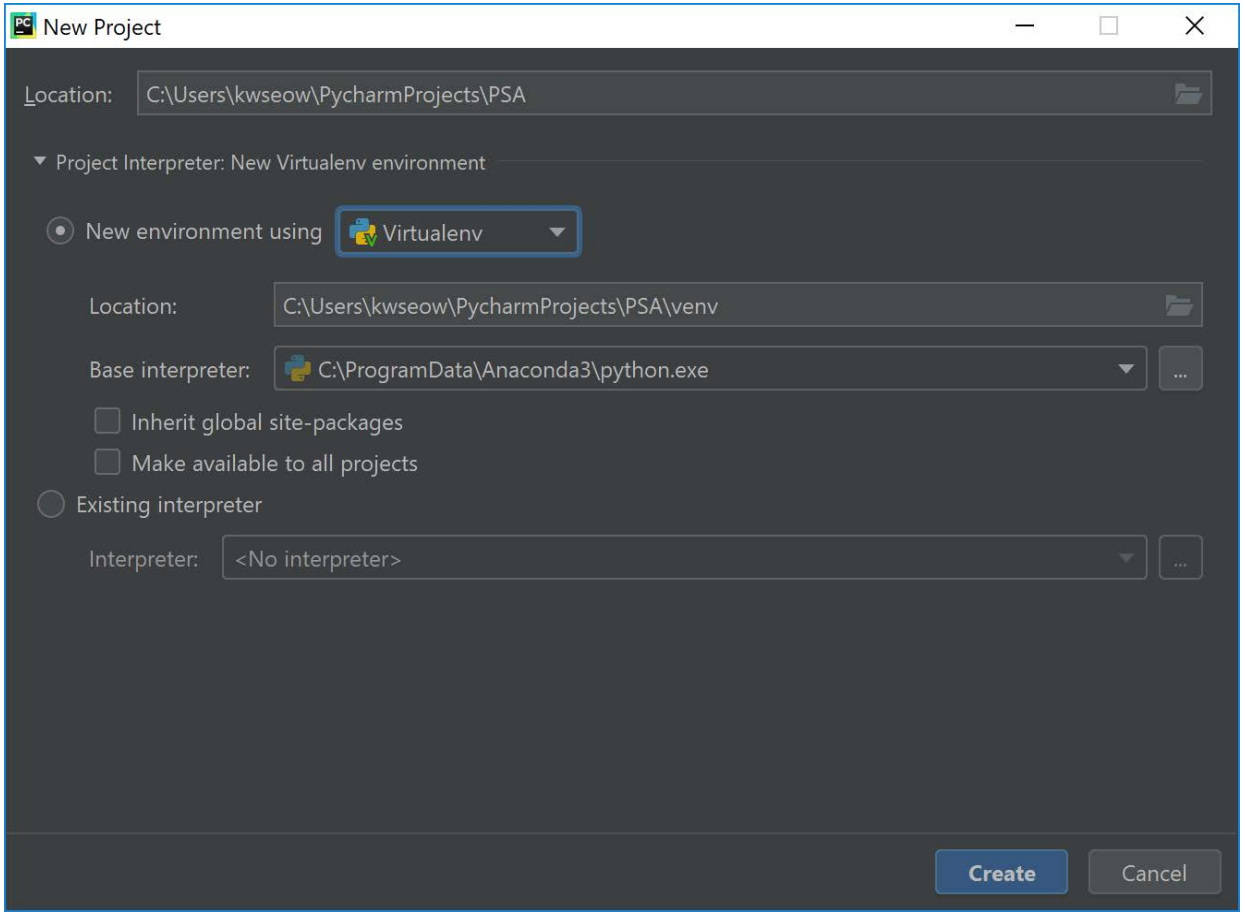
Use an IDE, Run PyCharm



Use an IDE, Run PyCharm

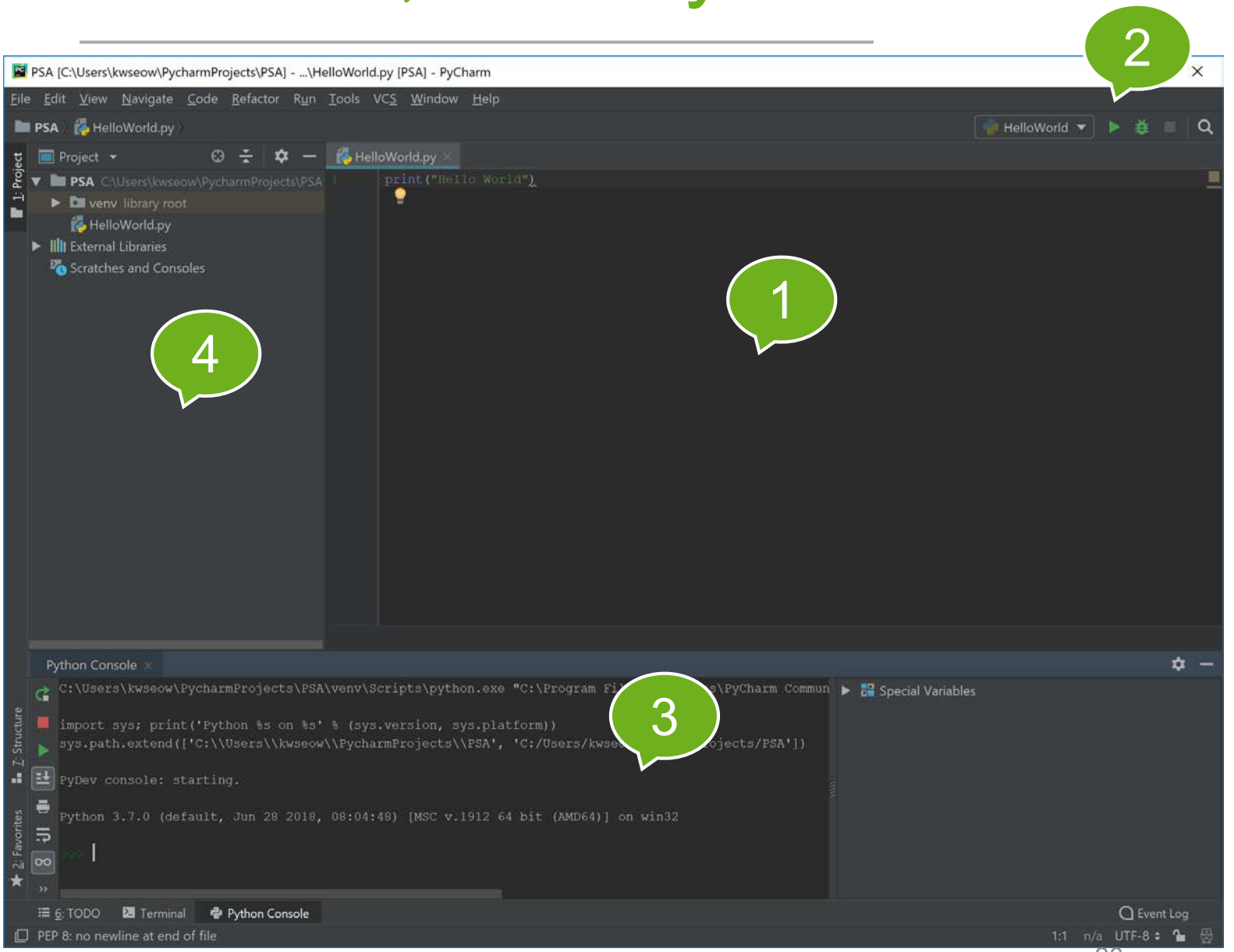
- Enter the project info

- 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)

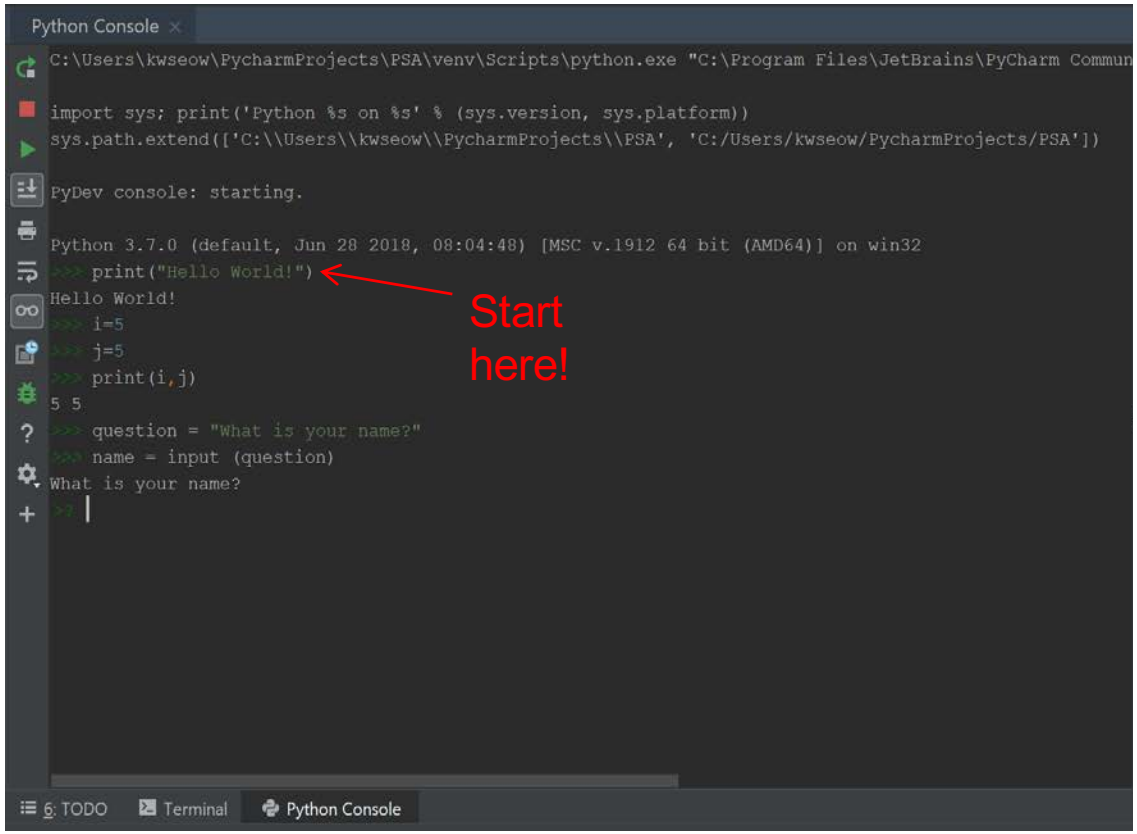

jetbrains.com/pycharm


blog.jetbrains.com/pycharm


@pycharm

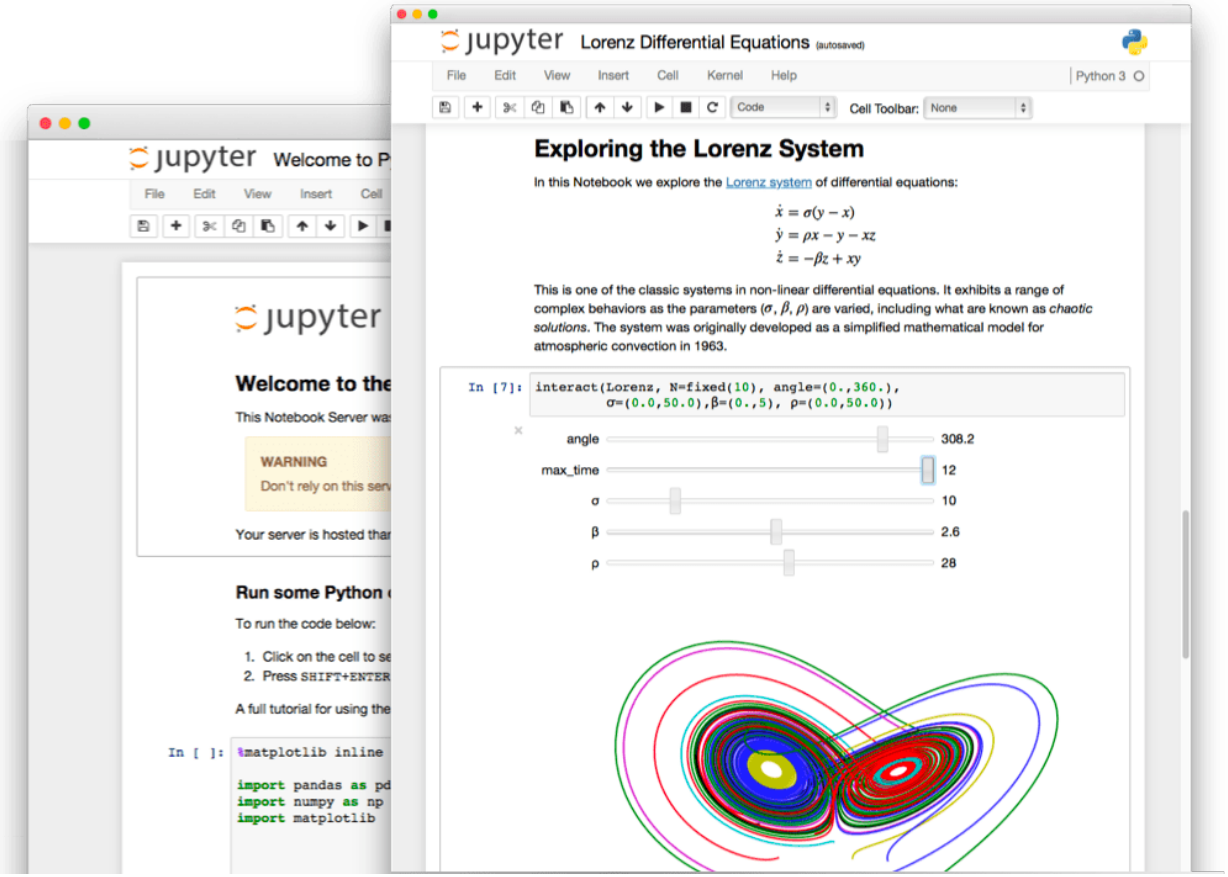
Using the console

- 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

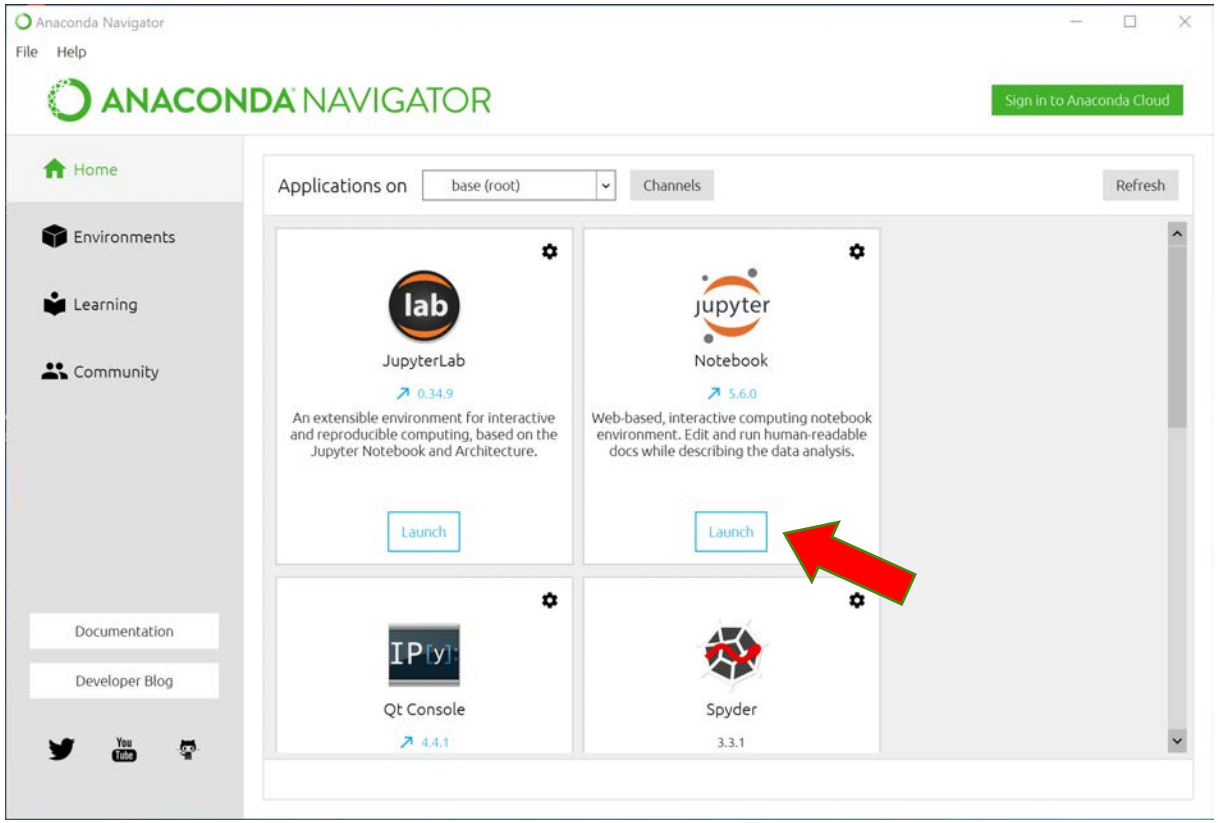
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning,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



Using Jyputer Notebook

Launch Anaconda Navigator from the start menu

Click on Jupyter Notebook to launch Jupyter

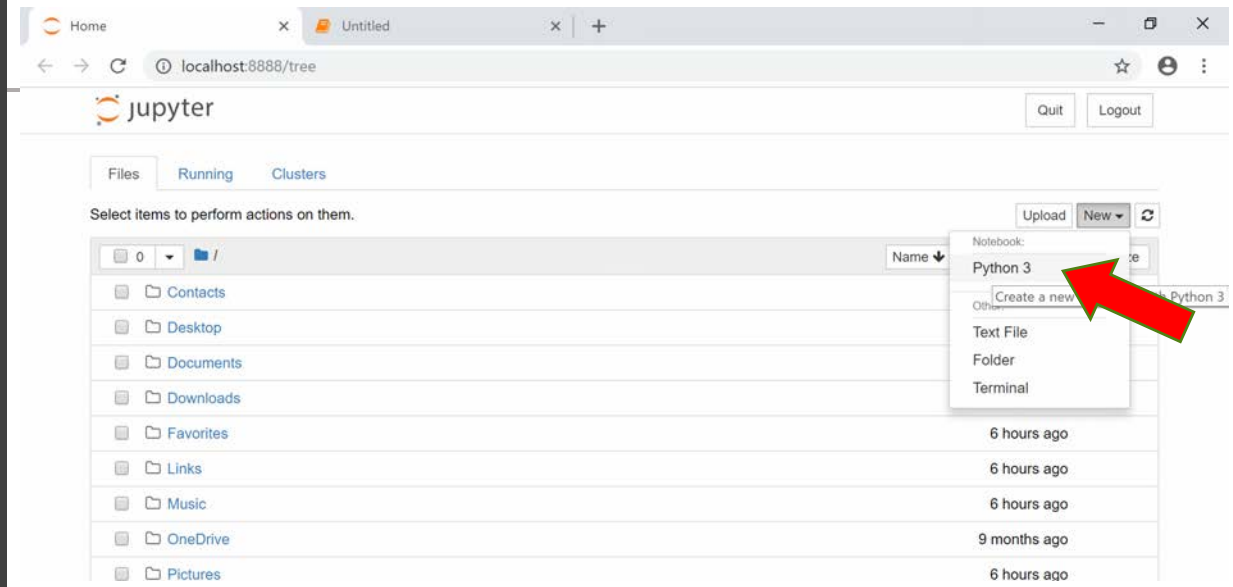


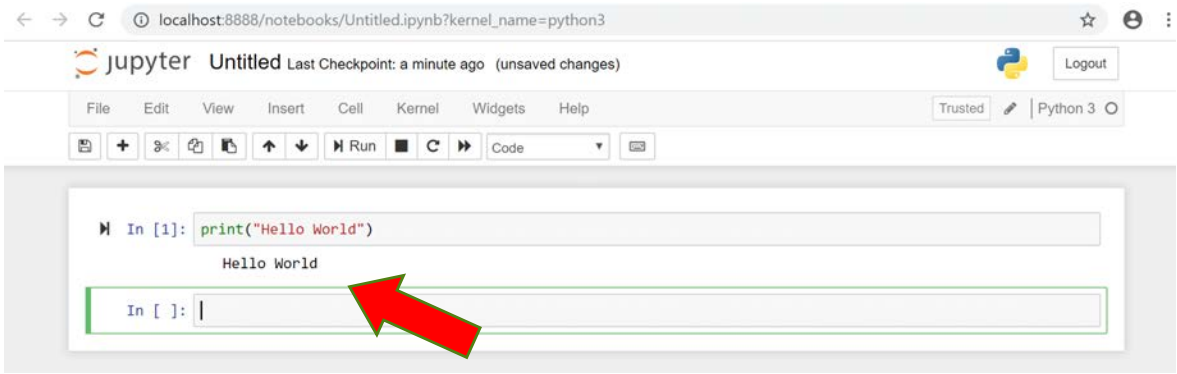
Using Jyputer Notebook

Notebook will launch with your default web browser.

Use New to start a new notebook

To run a “cell”, use shift-enter



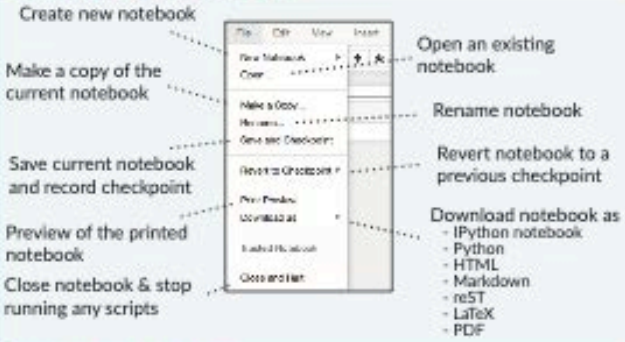


Python For Data Science *Cheat Sheet*

Jupyter Notebook

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

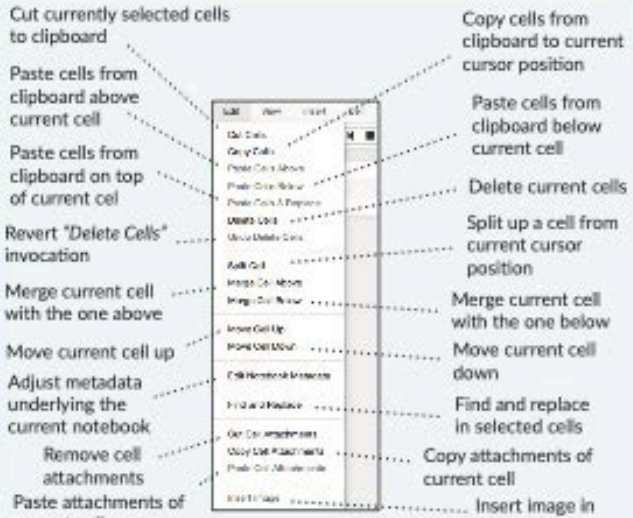
Saving/Loading Notebooks



Writing Code And Text

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

Edit Cells



Insert Cells

Working with Different Programming Languages

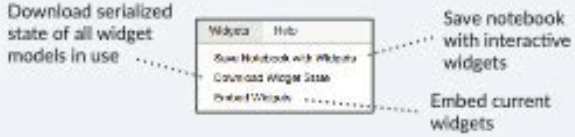
Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:

Installing Jupyter Notebook will automatically install the IPython kernel.

Widgets

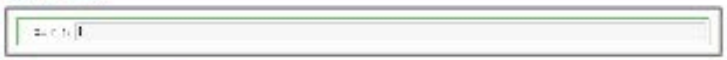
Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc.

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



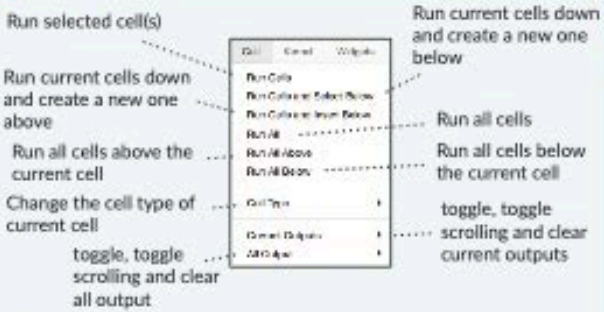
Command Mode:

Edit Mode:



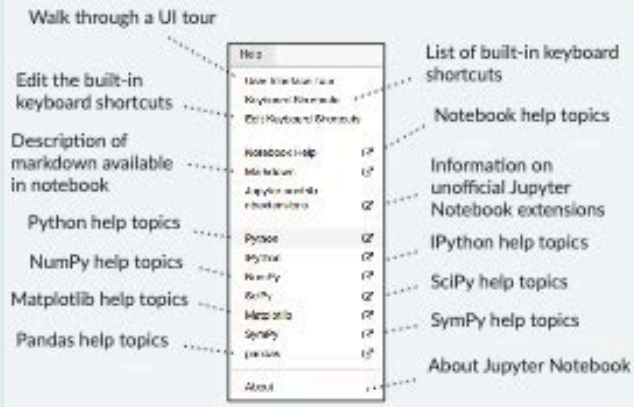
1. Save and checkpoint
2. Insert cell below
3. Cut cell
4. Copy cell(s)
5. Paste cell(s) below
6. Move cell up
7. Move cell down
8. Run current cell
9. Interrupt kernel
10. Restart kernel
11. Display characteristics
12. Open command palette
13. Current kernel
14. Kernel status
15. Log out from notebook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

Executing Cells



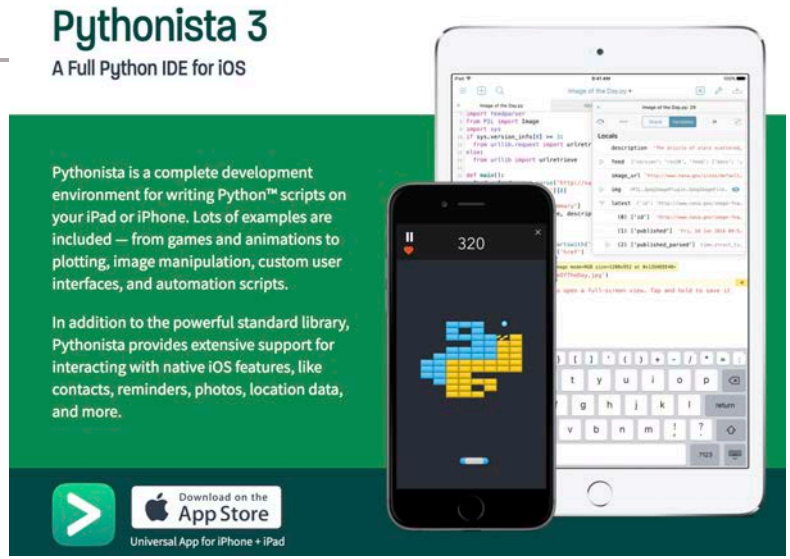
View Cells

Asking For Help



DataCamp
Learn Python for Data Science Interactively

http://omz-software.com/pythonista/



https://pybee.org/

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

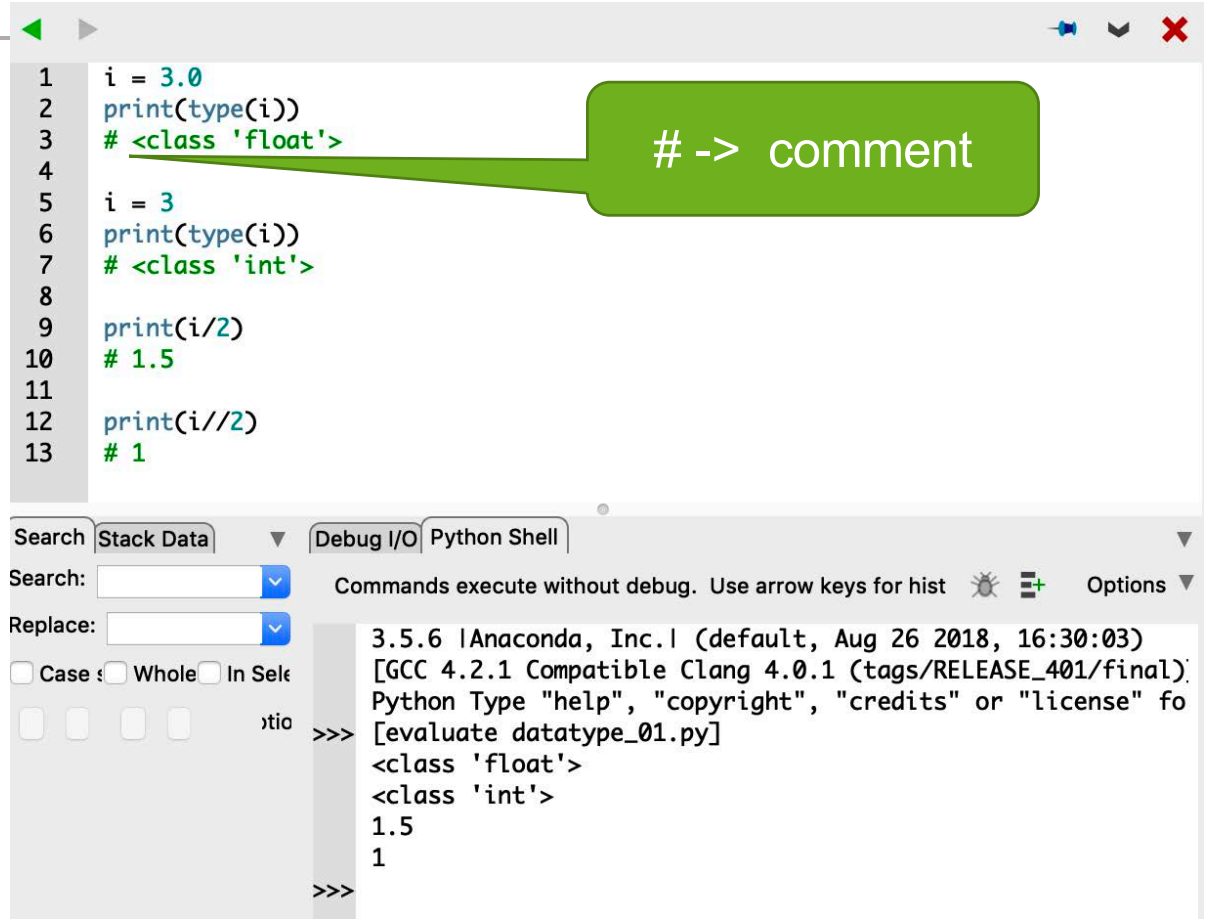
str	for a sequence of characters

Containers

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

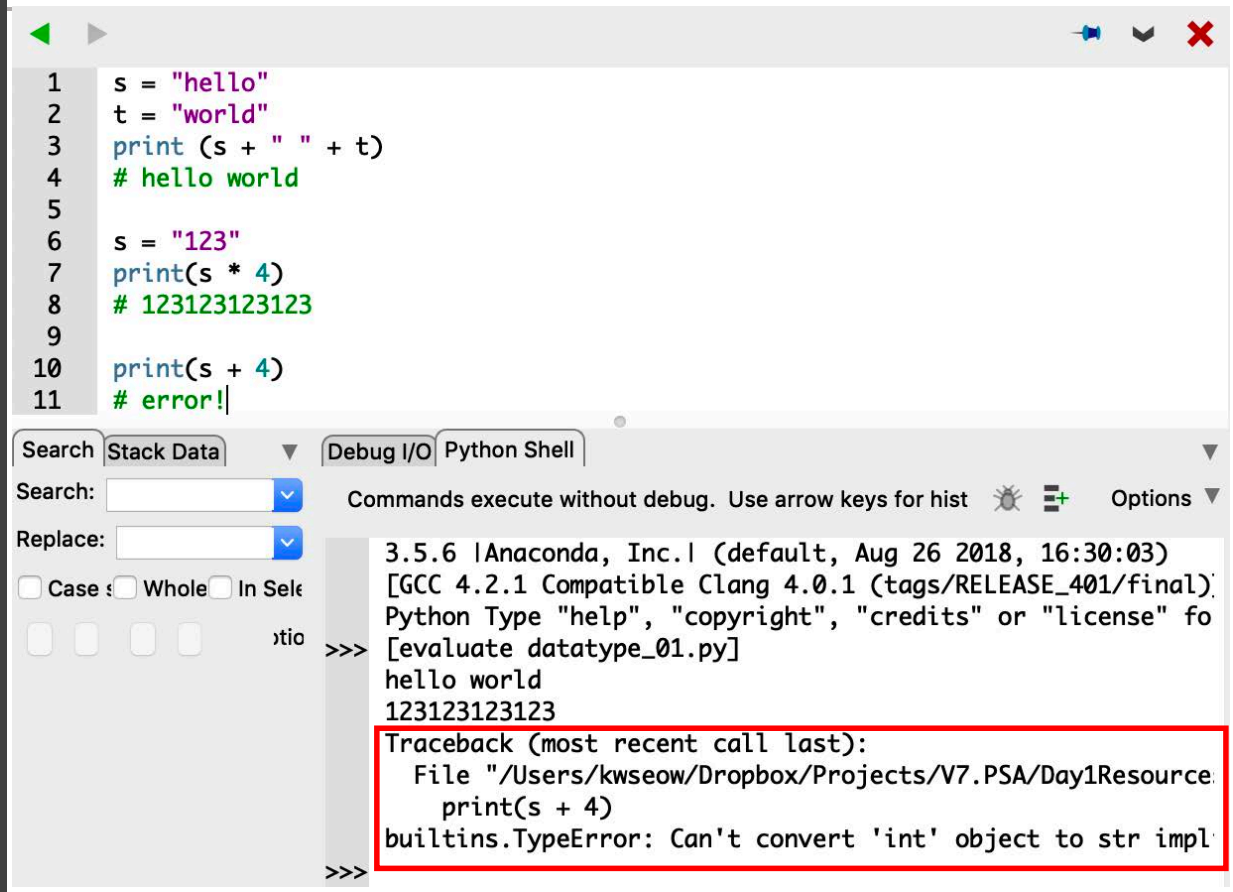
Data Types - Numbers

- 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.



Data Types - Strings

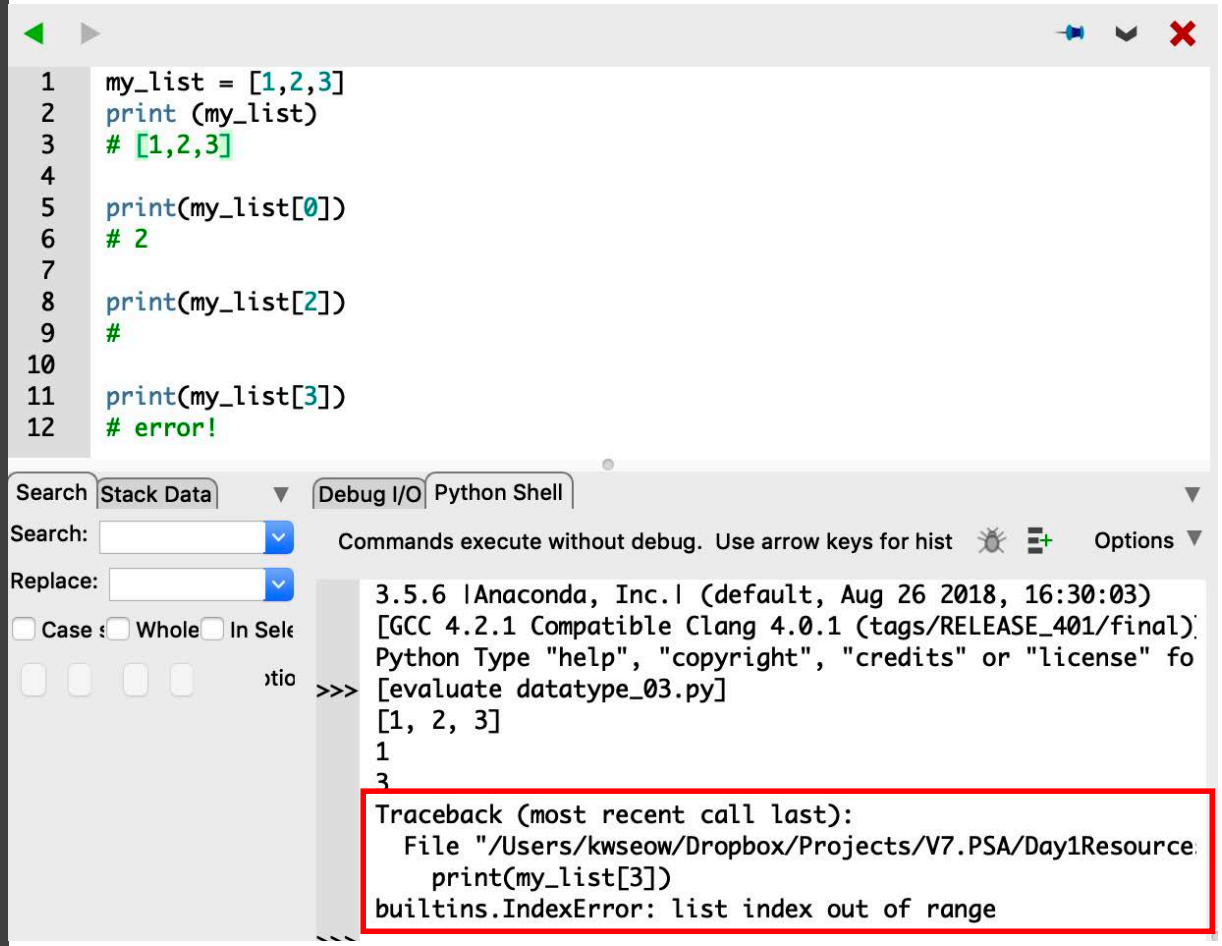
- Strings contain characters
- Strings can be added together with the + operator
- Strings can contain number characters, but they are not numbers (int or float)
- You can not add a number to a string
- The * operator will replicate the string with an integer value



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

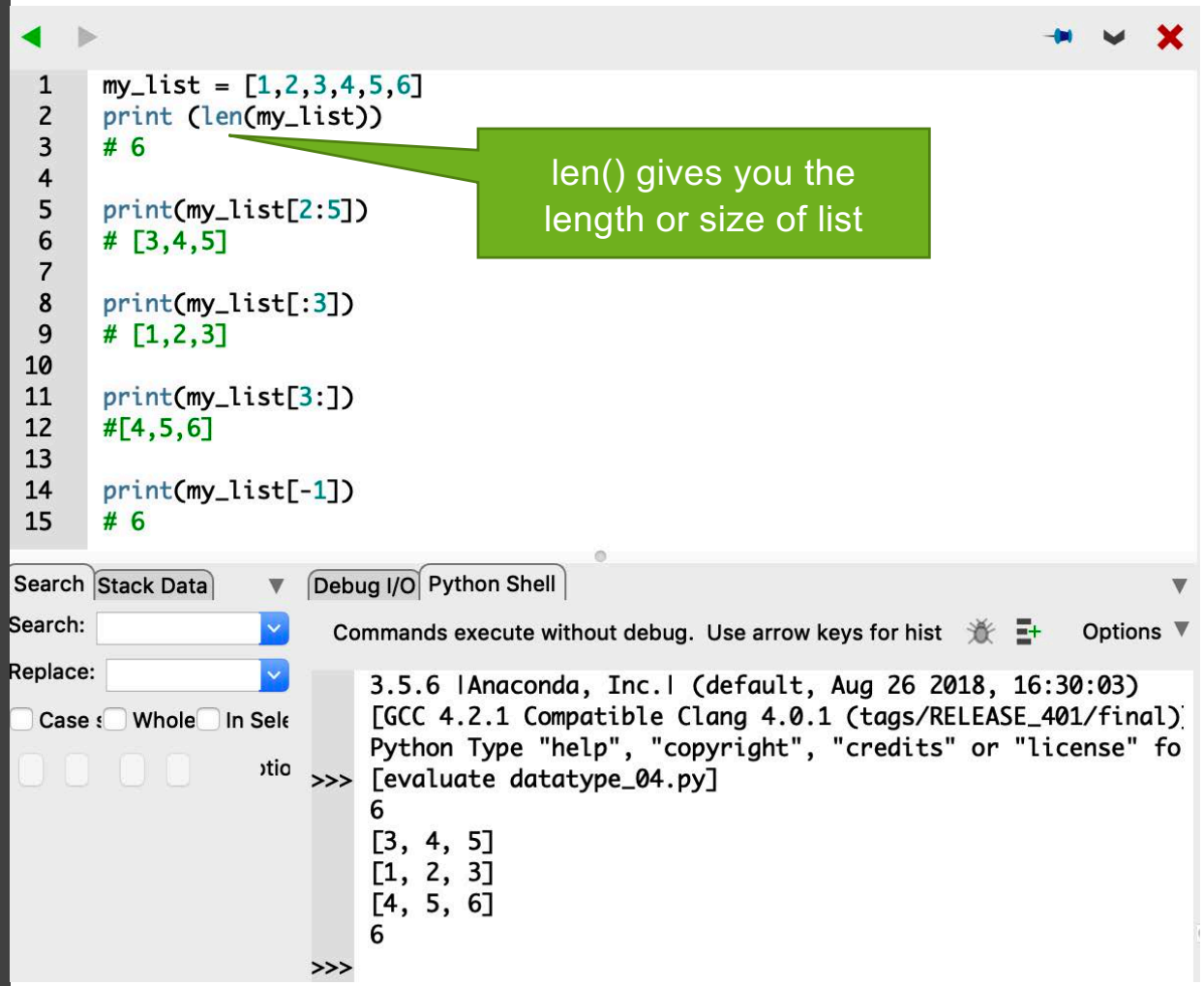
Data Types - List

- 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?



Data Types - List

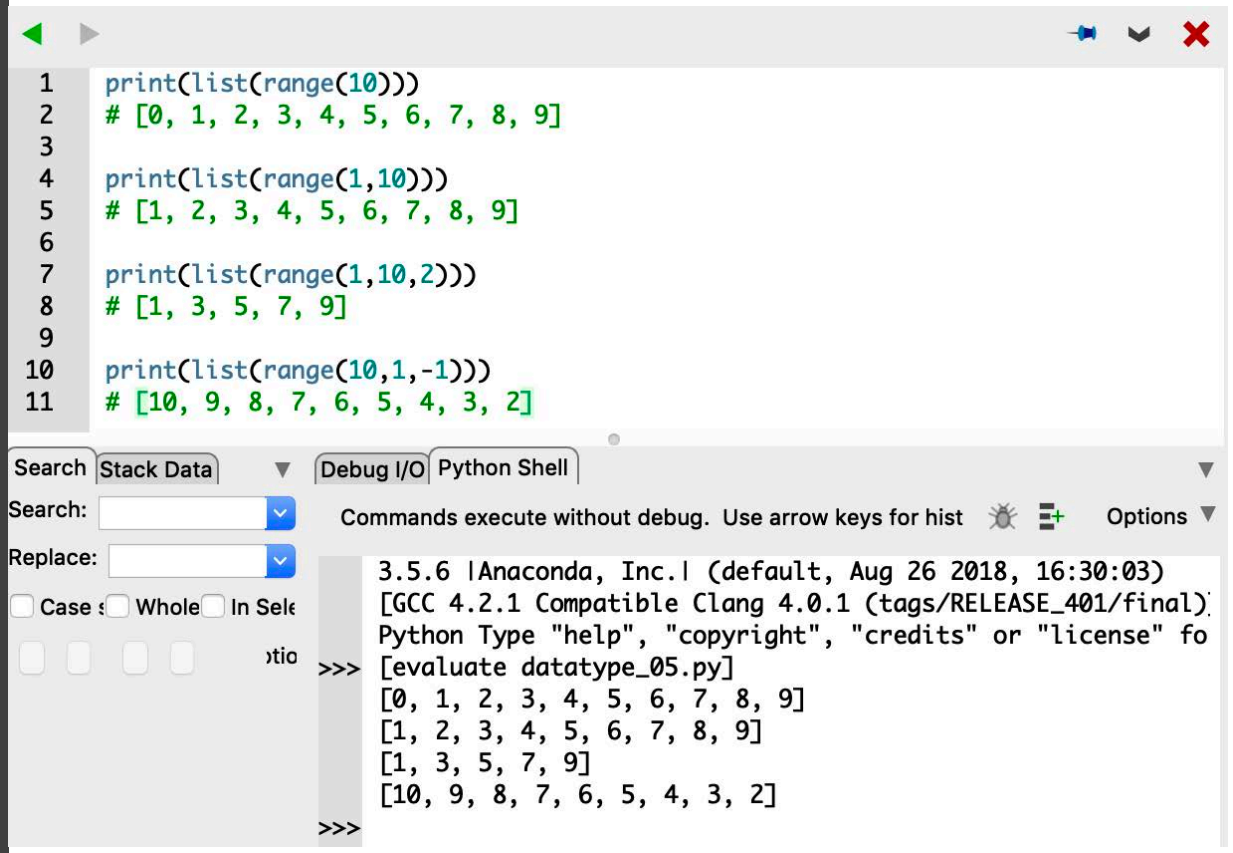
- len gives you the length or size of list
- Get a range of items using : colon
  (This is called slicing)
  - [:3] first 3 items
  - [3:] last 3 items
- Negative index gives you items from the back
  - [-x] x^{th} last item



Range

Three versions:

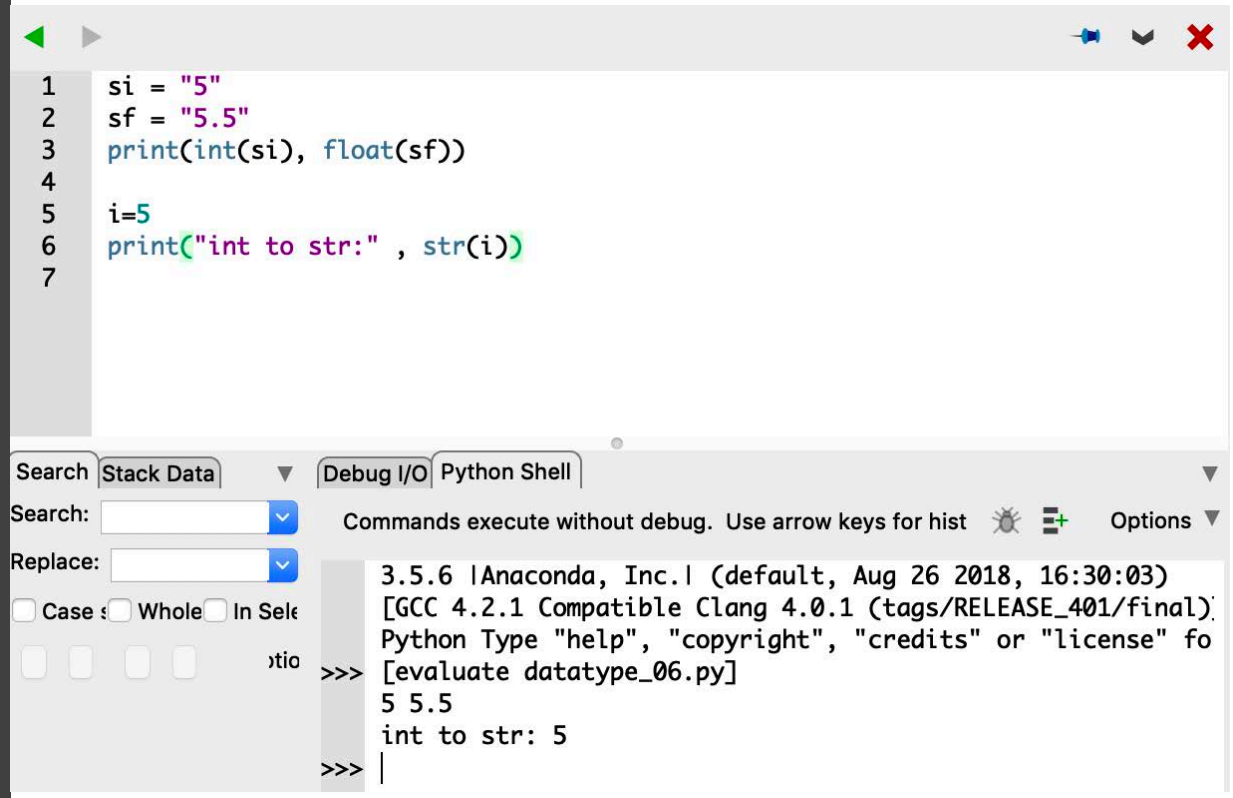
- `range(y)`
  - starts at 0
  - ends before `y`
  - step up by 1
- `range(x, y)`
  - starts at `x`
  - ends before `y`
  - step up by 1
- `range(x, y, s)`
  - starts at `x`
  - ends before `y`
  - step up by `s`



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(...)



Print Formatted Numbers

Formatting numbers

%d int
%f float
%s string

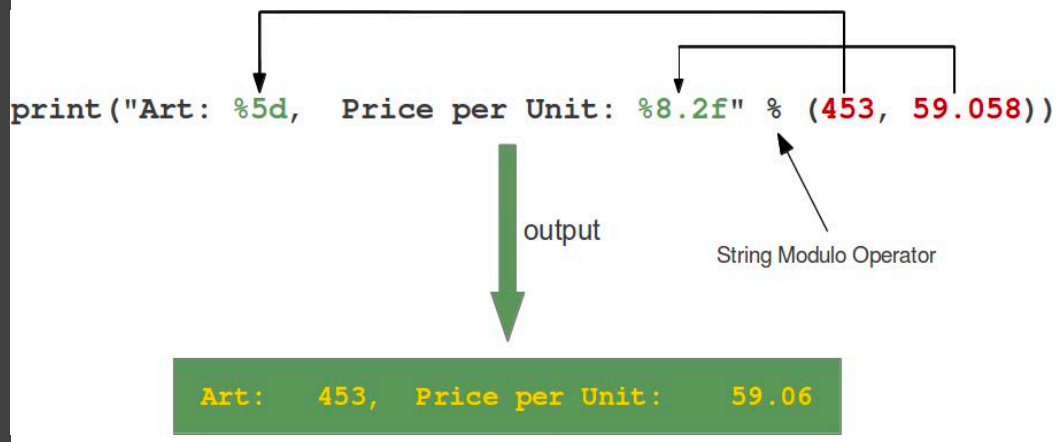
Special formatting

%.2f

- float
- two digits behind the point

%+d (or f)

- force print the sign




```
1   import math
2   print("Pi is " + str (math.pi))
3   # Pi is 3.141592653589793
4
5   print("Pi is approx %0.2f" % (math.pi))
6   # Pi is approx 3.14
7
8   print("Pos or Neg: %+d %+d" % (-5,3))
9   # Pos or Neg: -5 +3
10
11  tmp_str = "Pos or Neg"
12  print("%s: %+d %+d" % (tmp_str,-5,3))
13  # Pos or Neg: -5 +3
```

```
Search | Stack Data
Search:
Replace:
Case s  Whole  In Sele
Debug I/O | Python Shell
Commands execute without debug.  Use arrow keys for hist   Options
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)
Python Type "help", "copyright", "credits" or "license" fo
>>> [evaluate print_format_01.py]
Pi is 3.141592653589793
Pi is approx 3.14
Pos or Neg: -5 +3
Pos or Neg: -5 +3
>>>
```

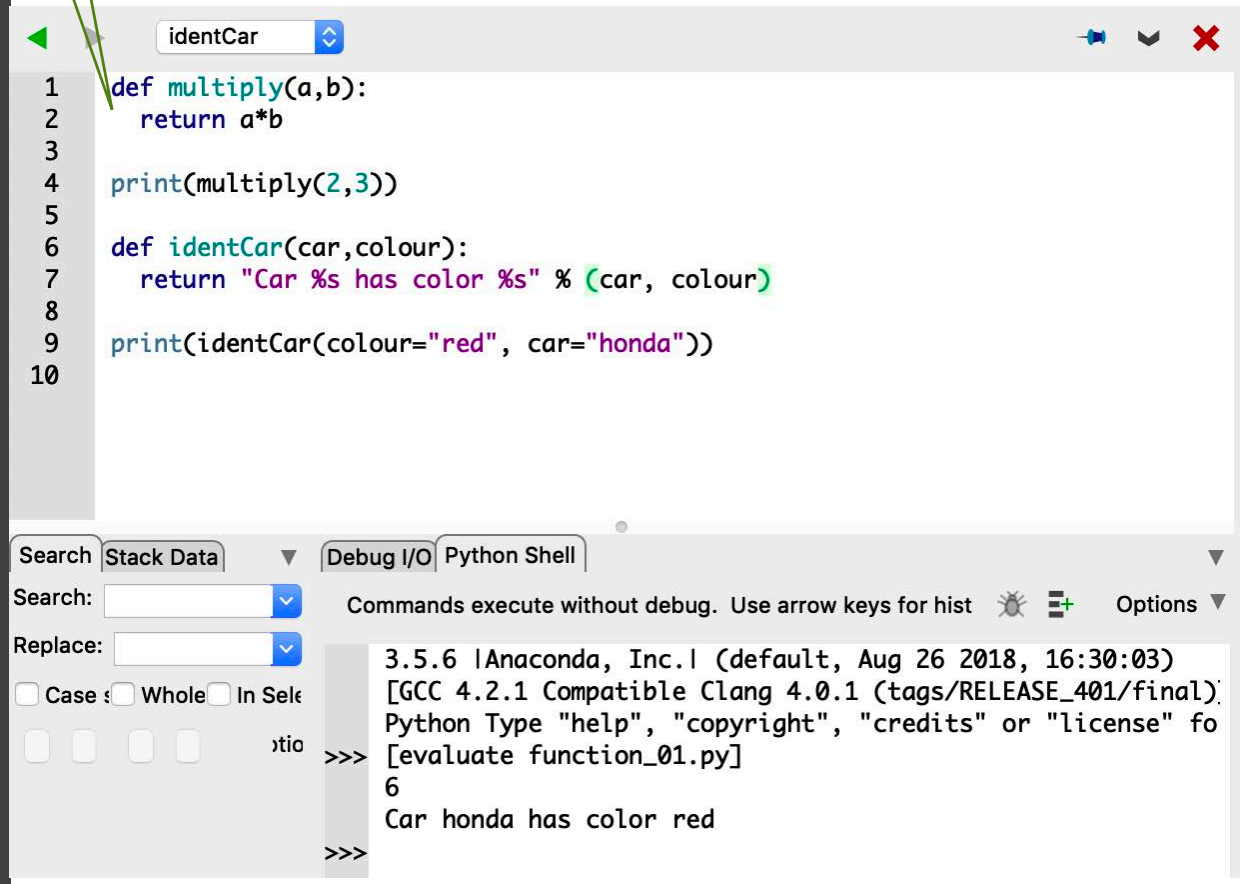
Functions

Passing parameters

Arguments must be passed in order

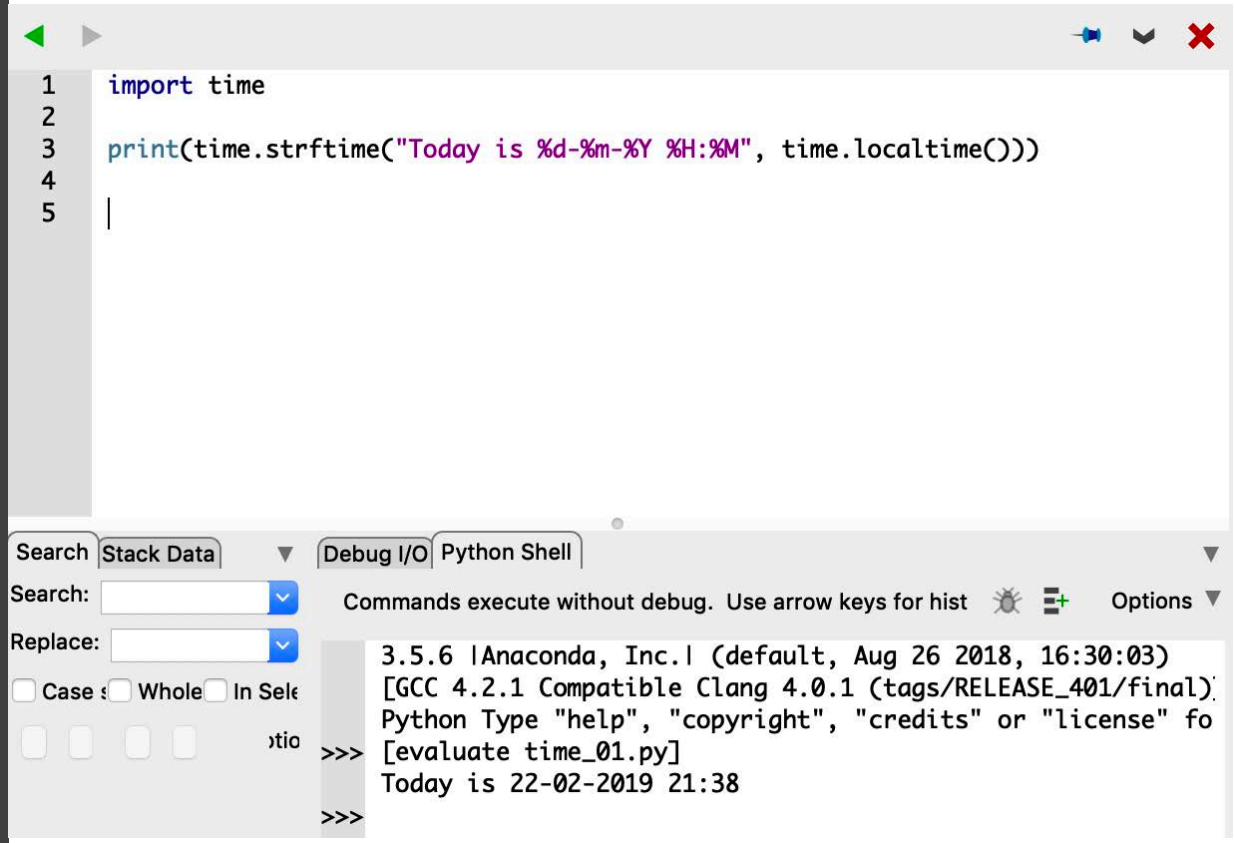
Alternatively, use parameter names to identify the arguments

Be sure to have indentation



The time library

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



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

The datetime library

The datetime module allows manipulating dates and times in both simple and complex ways. date and time arithmetic is supported

```python
from datetime import datetime, timedelta
d1 = datetime(1991, 4, 30)
print(d1)
# -> 1991-04-30 00:00:00

d2 = d1 + timedelta(10)
print(d2)
# -> 1991-05-10 00:00:00

print(d2 - d1)
# -> 10 days, 0:00:00

d3 = d1 - timedelta(100)
print(d3)
# -> 1991-01-20 00:00:00

d4 = d1 - 2 * timedelta(50)
print(d4)

#adding seconds
d1 = datetime(1991, 4, 30)
print(d1)
# -> 1991-04-30 00:00:00

d2 = d1 + timedelta(10,100)
print(d2)
# -> 1991-05-10 00:01:40

print(d2 - d1)
# -> 10 days, 0:01:40

#get current Date
d1 = datetime.now()
print(d1)
# -> 2019-03-07 12:51:51.196327
print(d1.day, d1.hour, d1.second)
# -> 2019-03-07 12:51:51.196327
```

https://docs.python.org/3/library/datetime.html

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

```
1  mick = 3.5
2  alice = 2.5
3  total = (mick + alice) * 60 * 60
4  print("Mick took ", mick, "hr")
5  print("Alice took", alice, "hr")
6  print("Total time is", total, "s")
7
8
```

Search | Stack Data | Debug I/O | Python Shell

Search:

Replace:

Case s Whole In Sele

Commands execute without debug. Use arrow keys for hist Options

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)
Python Type "help", "copyright", "credits" or "license" fo
>>> [evaluate Calculator.py]
Mick took  3.5 hr
Alice took 2.5 hr
Total time is 21600.0 s
>>>
```

Exercise
Time Conversion

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

```
1  myTime = 1000
2  myMins = myTime // 60
3  mySecs = myTime % 60
4  print(myMins,"mins &", mySecs,"sec")
5
```

Search | Stack Data | Debug I/O | Python Shell

Search:

Replace:

Case s Whole In Sele

Commands execute without debug. Use arrow keys for hist Options

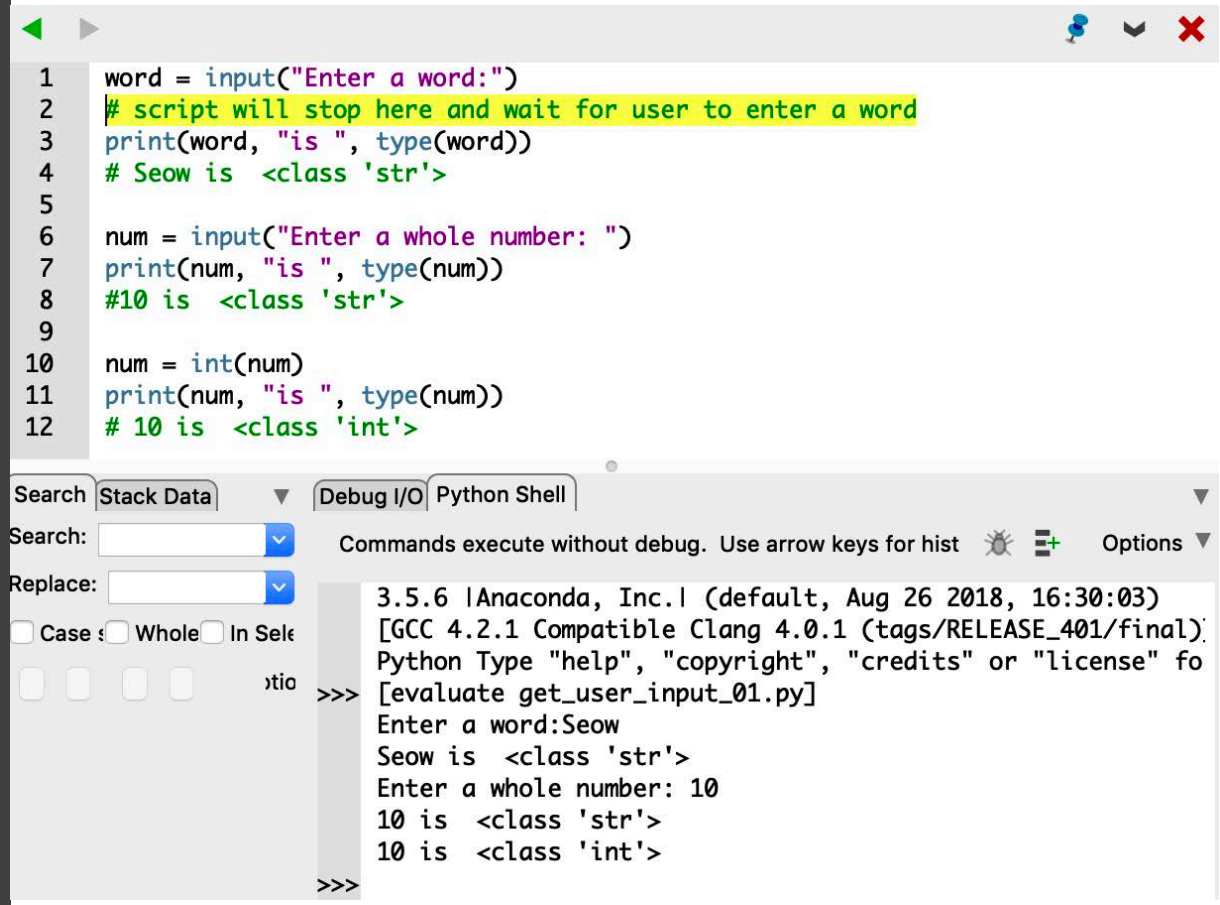
```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)
Python Type "help", "copyright", "credits" or "license" fo
>>> [evaluate TimeConversion.py]
16 mins & 40 sec
>>>
```

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.



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.

```
1  name = input("Enter patient's name:")
2  temperature = float(input("Enter patient's temperature:"))
3  norm_temp=36.9
4  print("%s's temperature is %.1f degree celsius from %.1f." %(name,temperature-norm_temp, norm_temp))
```

Search | Stack Data

Search:

Replace:

Case ser | Whole w | In Select

Debug I/O | Python Shell

Commands execute without debug. Use arrow keys for history. Options

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" for more information
>>> [evaluate TemperatureCalculator.py]
Enter patient's name:Seow
Enter patient's temperature:38.7
Seow's temperature is 1.8 degree celsius from 36.9.
>>>
```

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`?

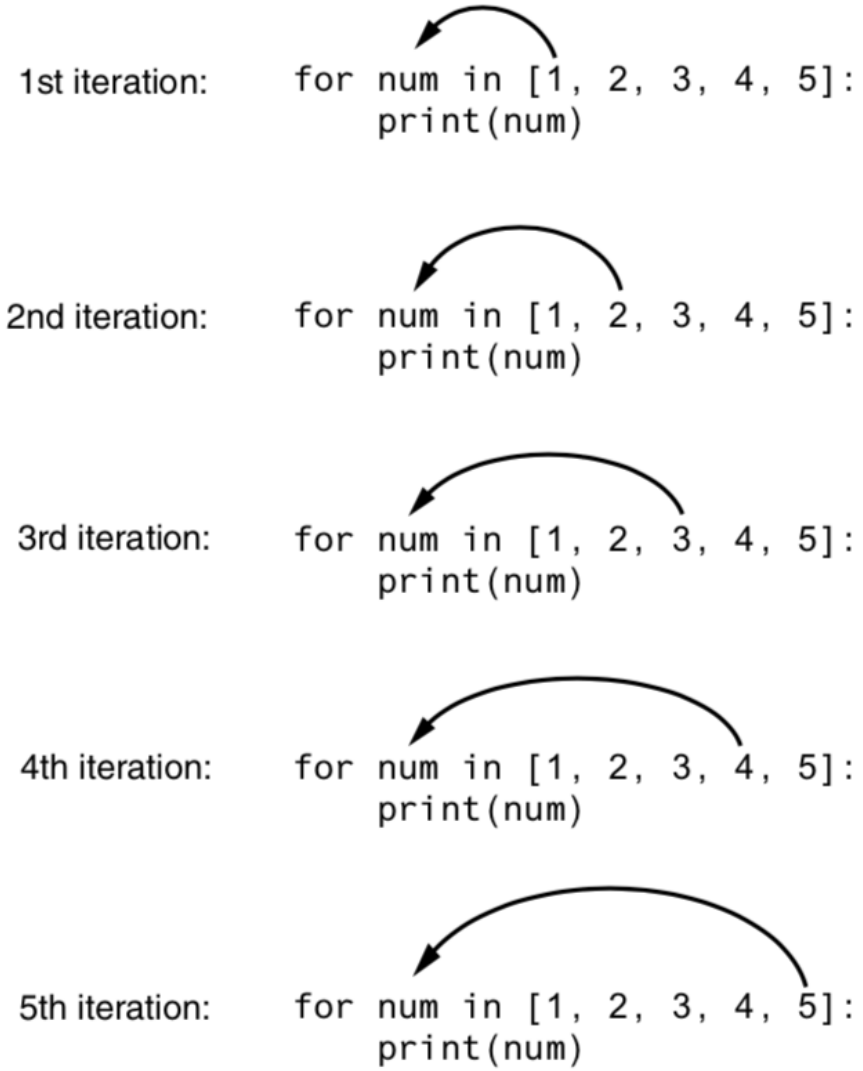


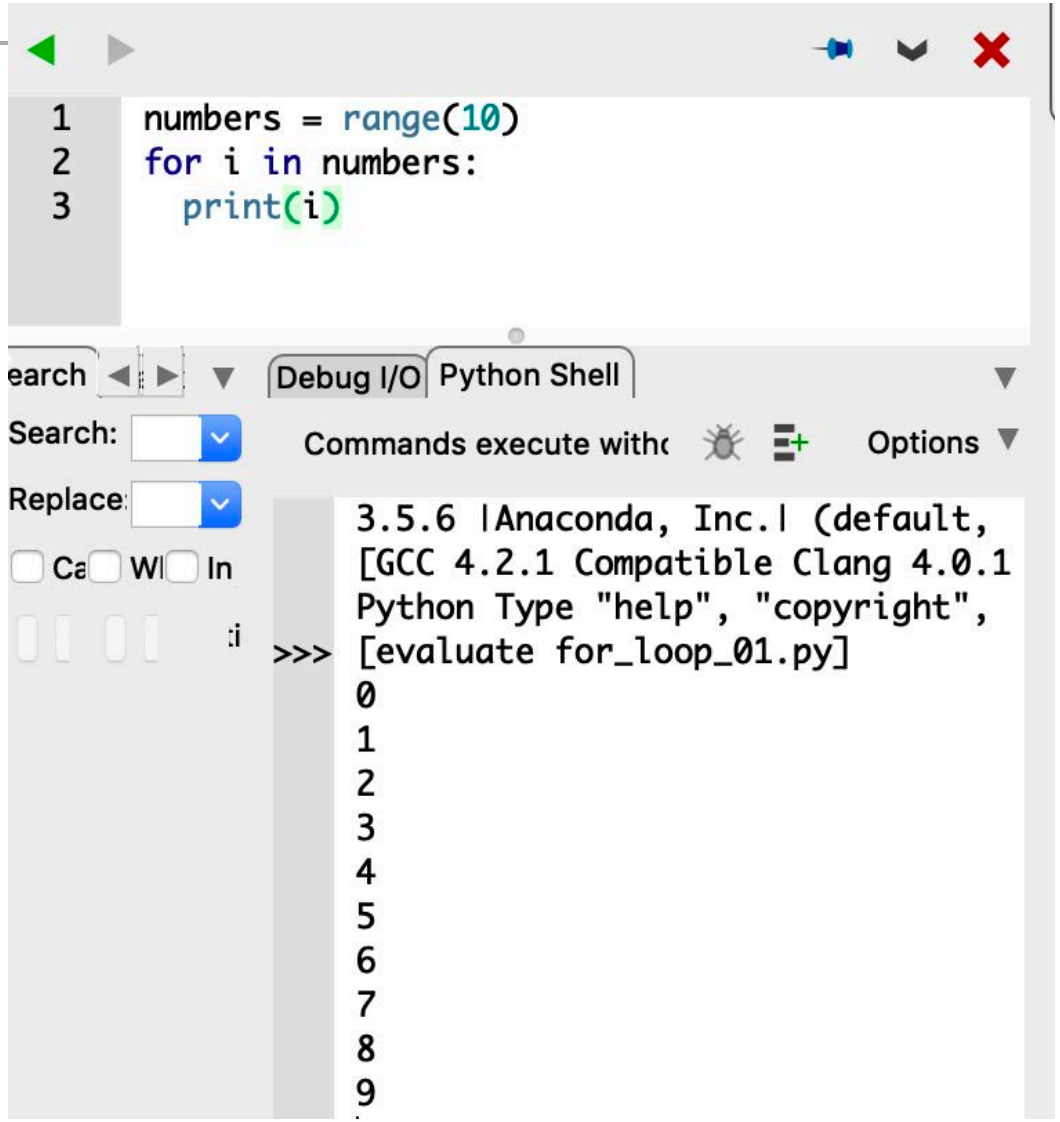
Image source : Starting out with Python, 4th Ed, Tony Gaddis

For Loops

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)?

Hint: print(i,end="")

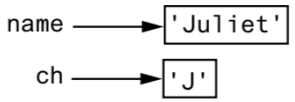


For Loops with string

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

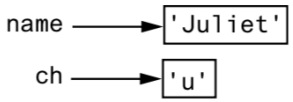
1st Iteration

```
for ch in name:
    print(ch)
```



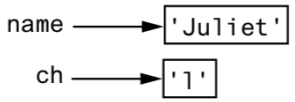
2nd Iteration

```
for ch in name:
    print(ch)
```



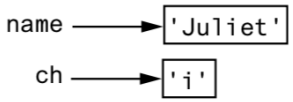
3rd Iteration

```
for ch in name:
    print(ch)
```



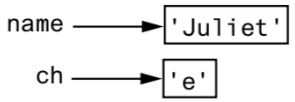
4th Iteration

```
for ch in name:
    print(ch)
```



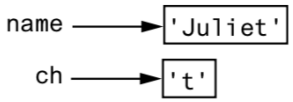
5th Iteration

```
for ch in name:
    print(ch)
```



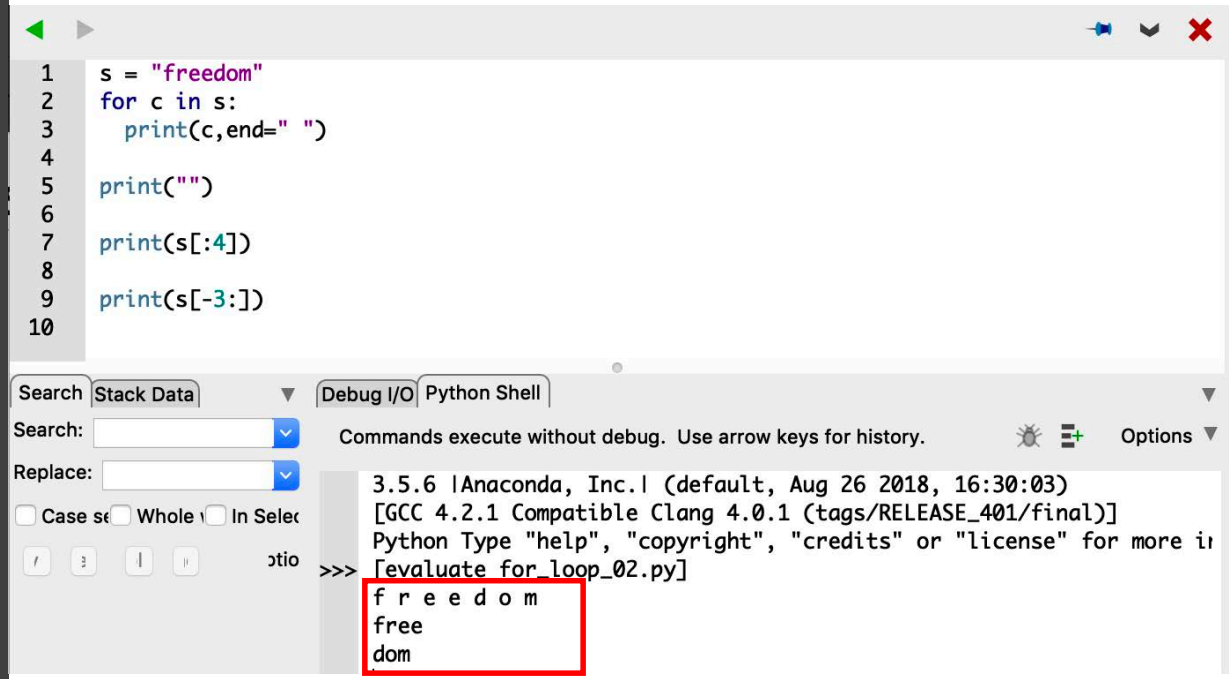
6th Iteration

```
for ch in name:
    print(ch)
```



For Loops with string

- 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.

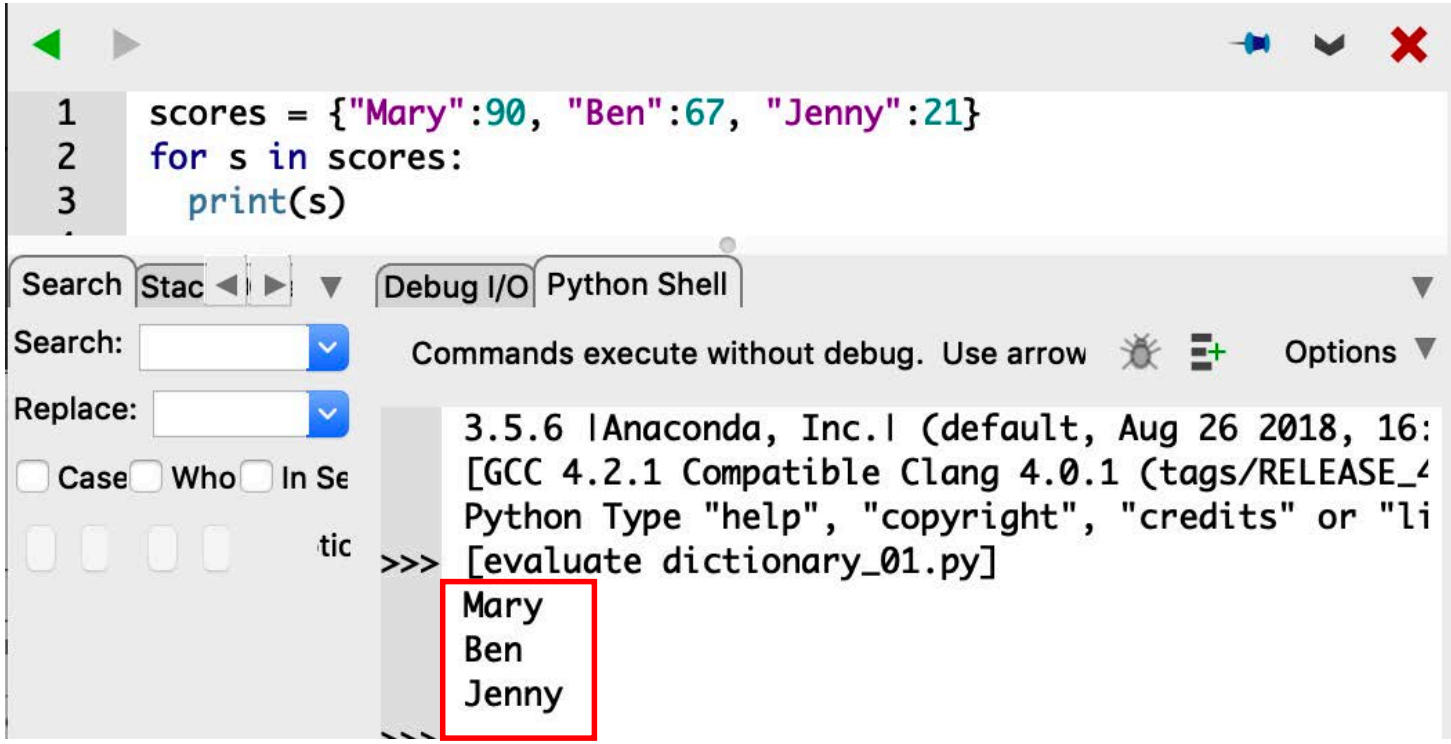


Data Types - Dictionary

- 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



- 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

A function is a group of statements that exist within a program for the purpose of performing a specific task.

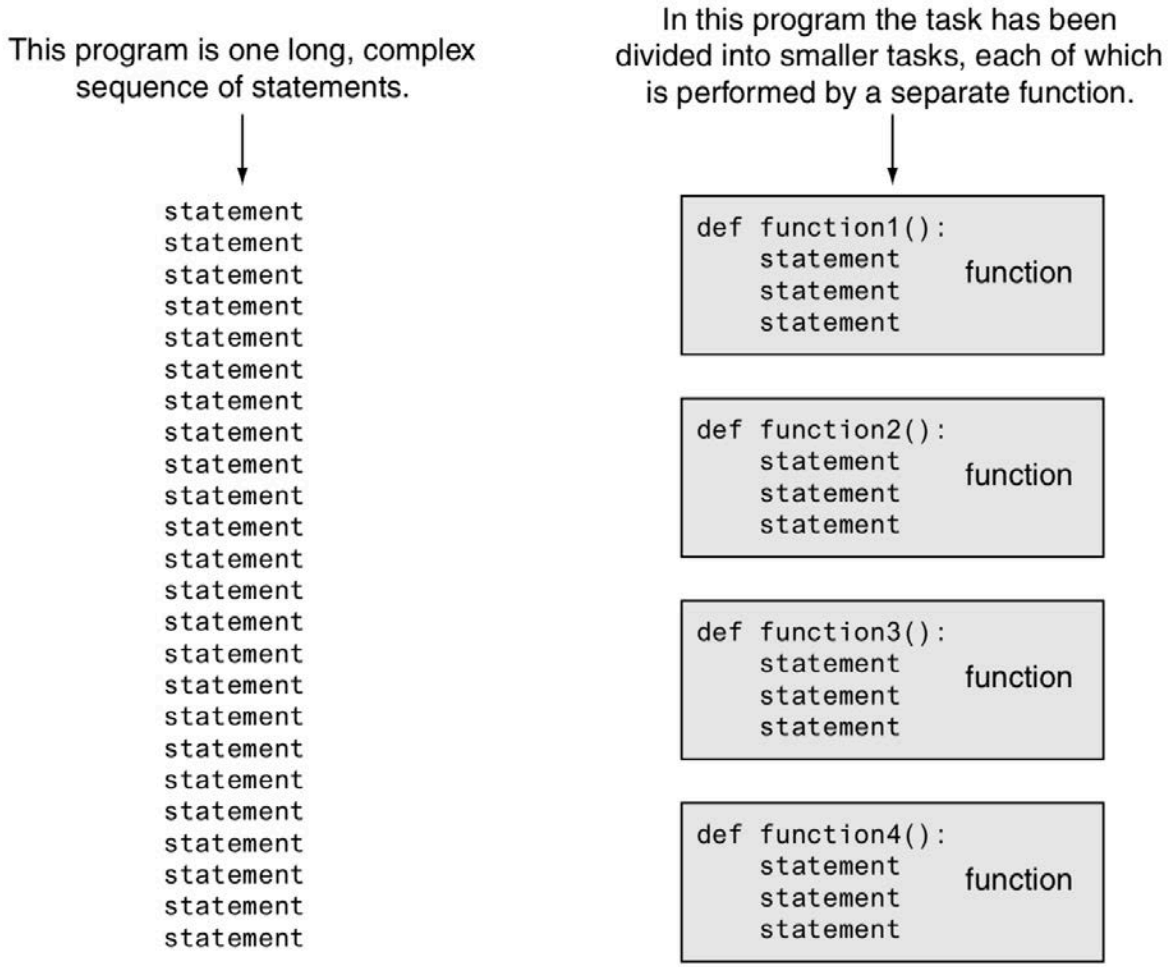


Image source : Starting out with Python, 4th Ed, Tony Gaddis

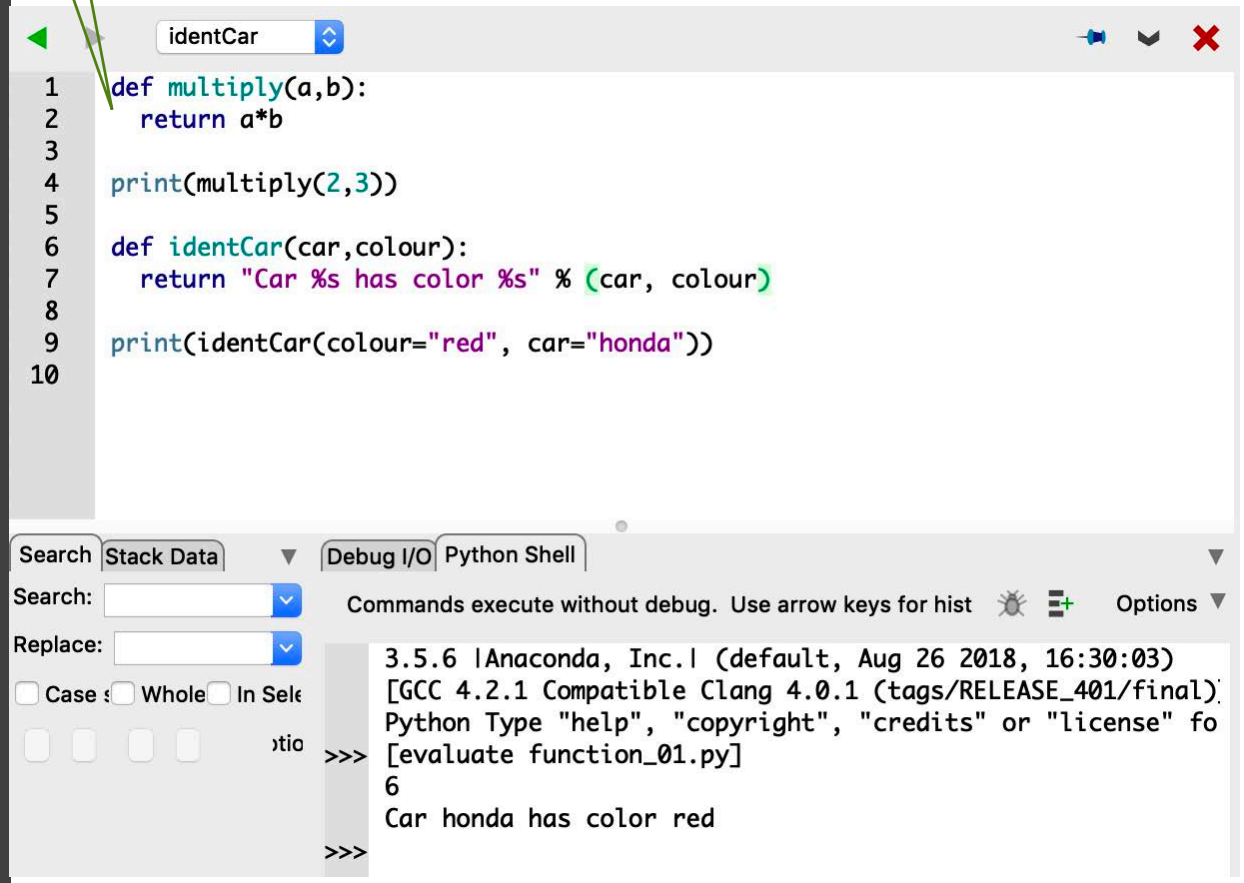
Functions

Passing parameters

- Arguments must be passed in order

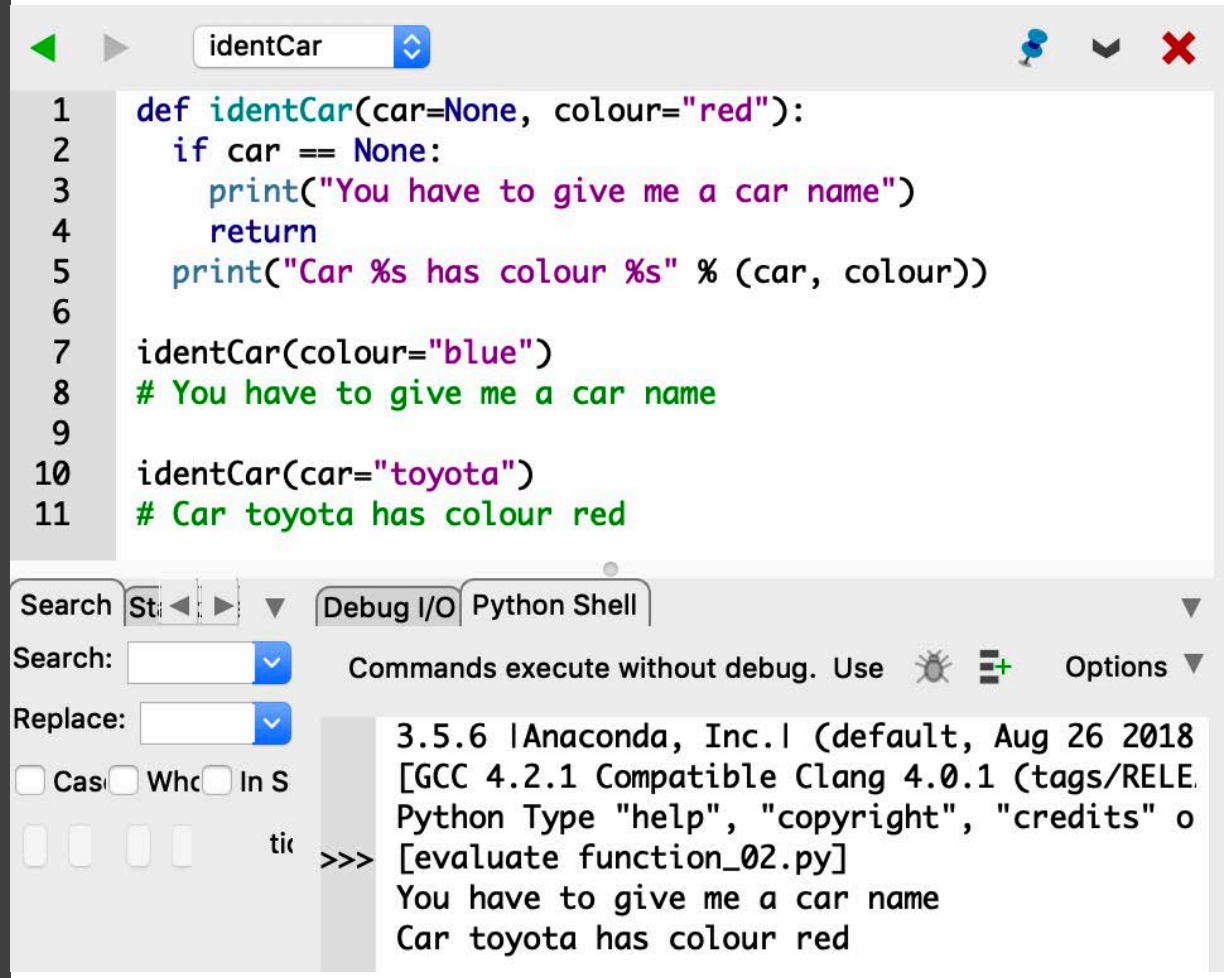
Alternatively, use parameter names to identify the arguments

Be sure to have indentation



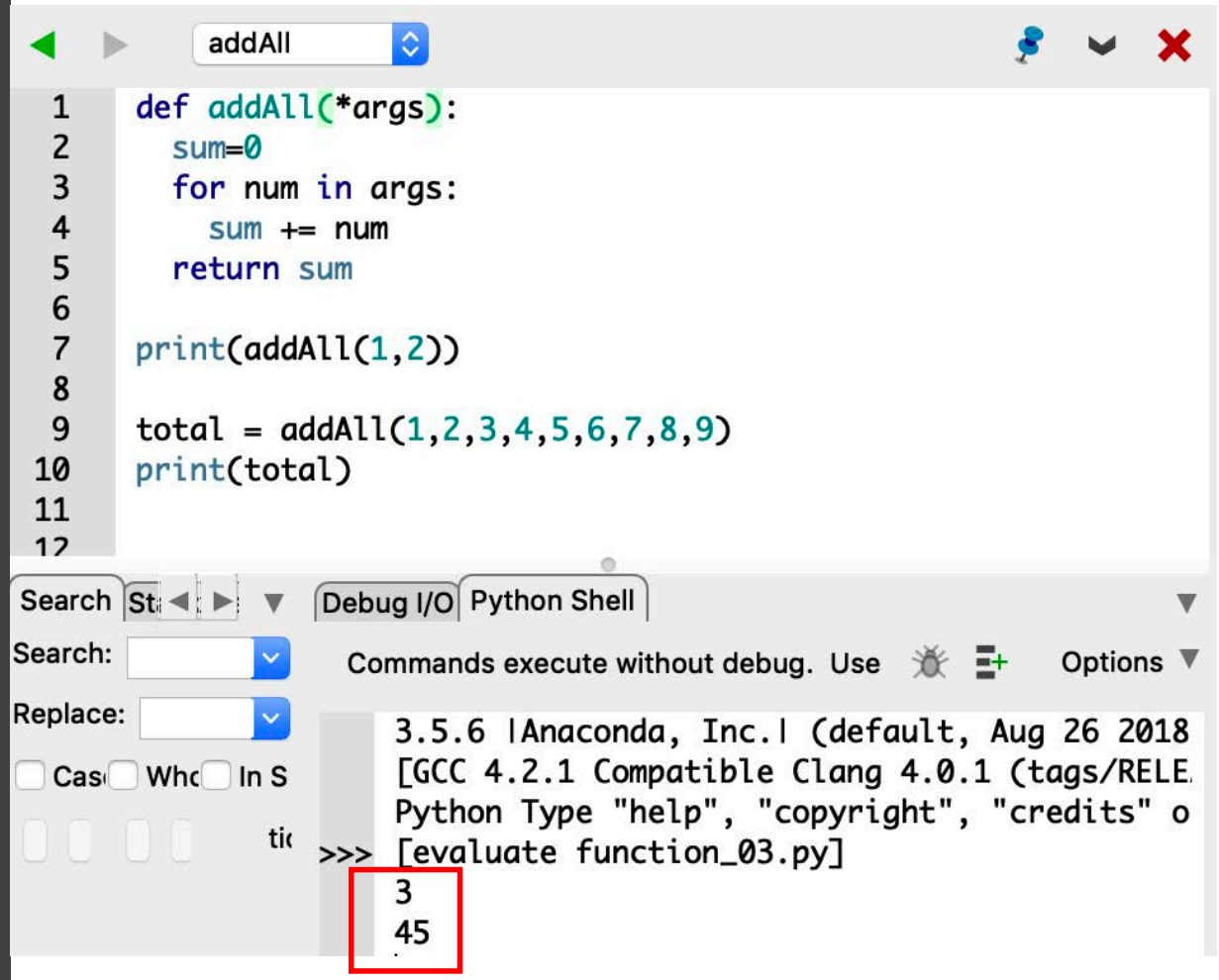
Functions - Default Parameters

Default parameters values and checking if parameter has been passed



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.



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

```
1  try:
2    5/0
3  except:
4    print("error")
5
6
```

```
Search  St
Search:
Replace:
Cas  Whc  In S
ti
```

```
Debug I/O  Python Shell
Commands execute without debug.  Use
Options
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELE
Python Type "help", "copyright", "credits" o
>>> [evaluate try_exception_01.py]
error
>>>
```

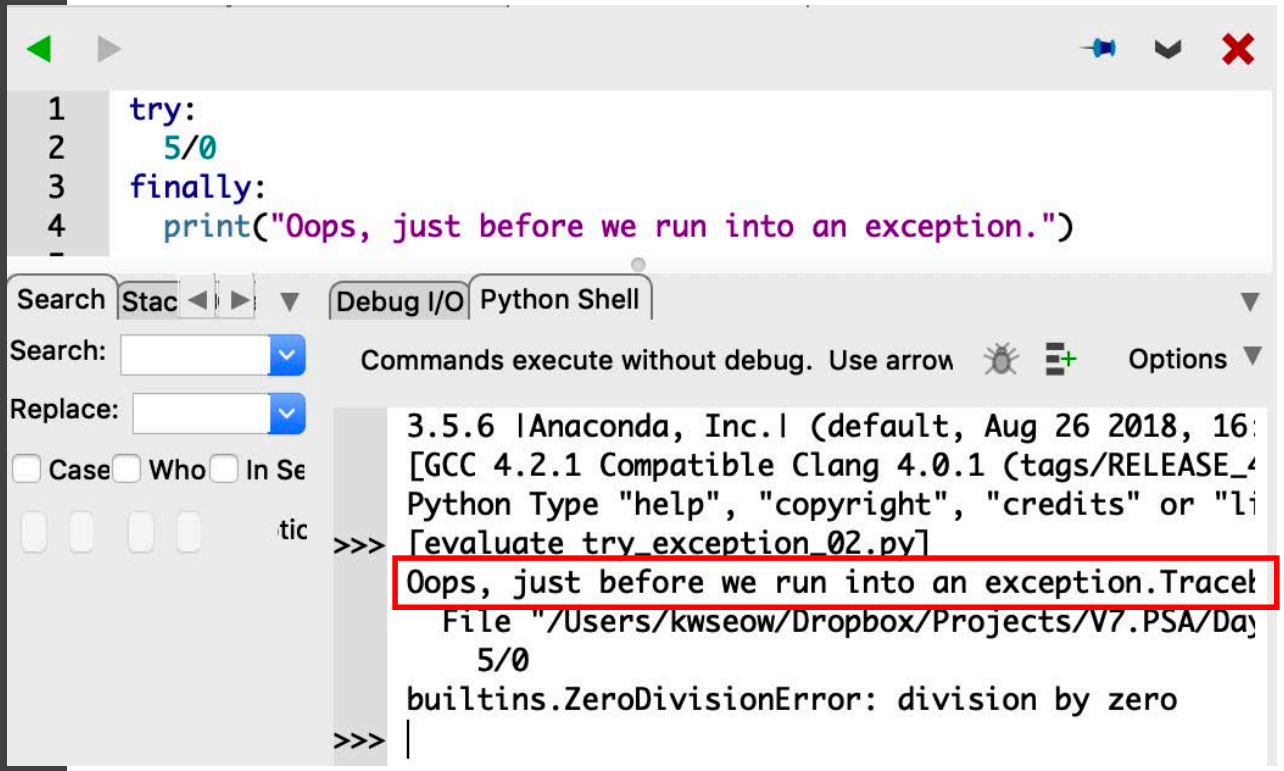
```
1  try:
2    5/0
3  except Exception as e:
4    print("Exception ", type(e) ,": " , e.args)
5
6
```

```
Search  Stack Data
Search:
Replace:
Case se  Whole  In Selec
otio
```

```
Debug I/O  Python Shell
Commands execute without debug.  Use arrow keys for history.
Options
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" for more i
>>> [evaluate try_exception_02.py]
Exception  <class 'ZeroDivisionError'> :  ('division by zero',)
```

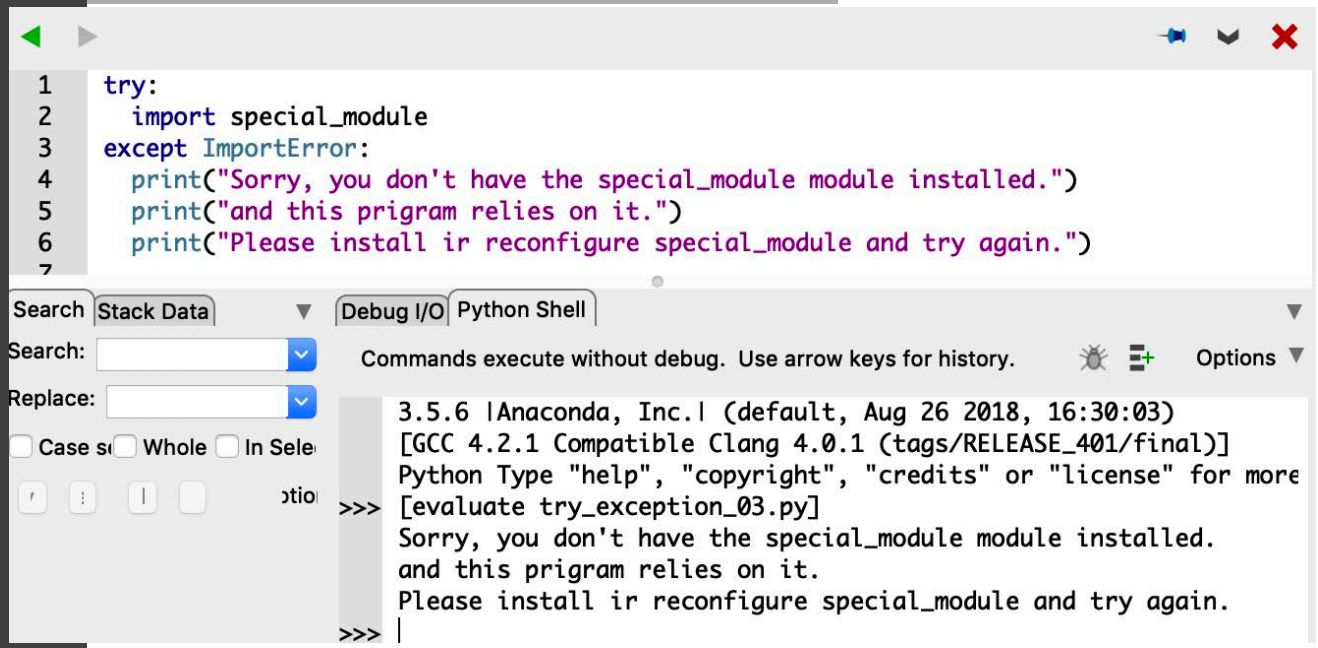
try .. except

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



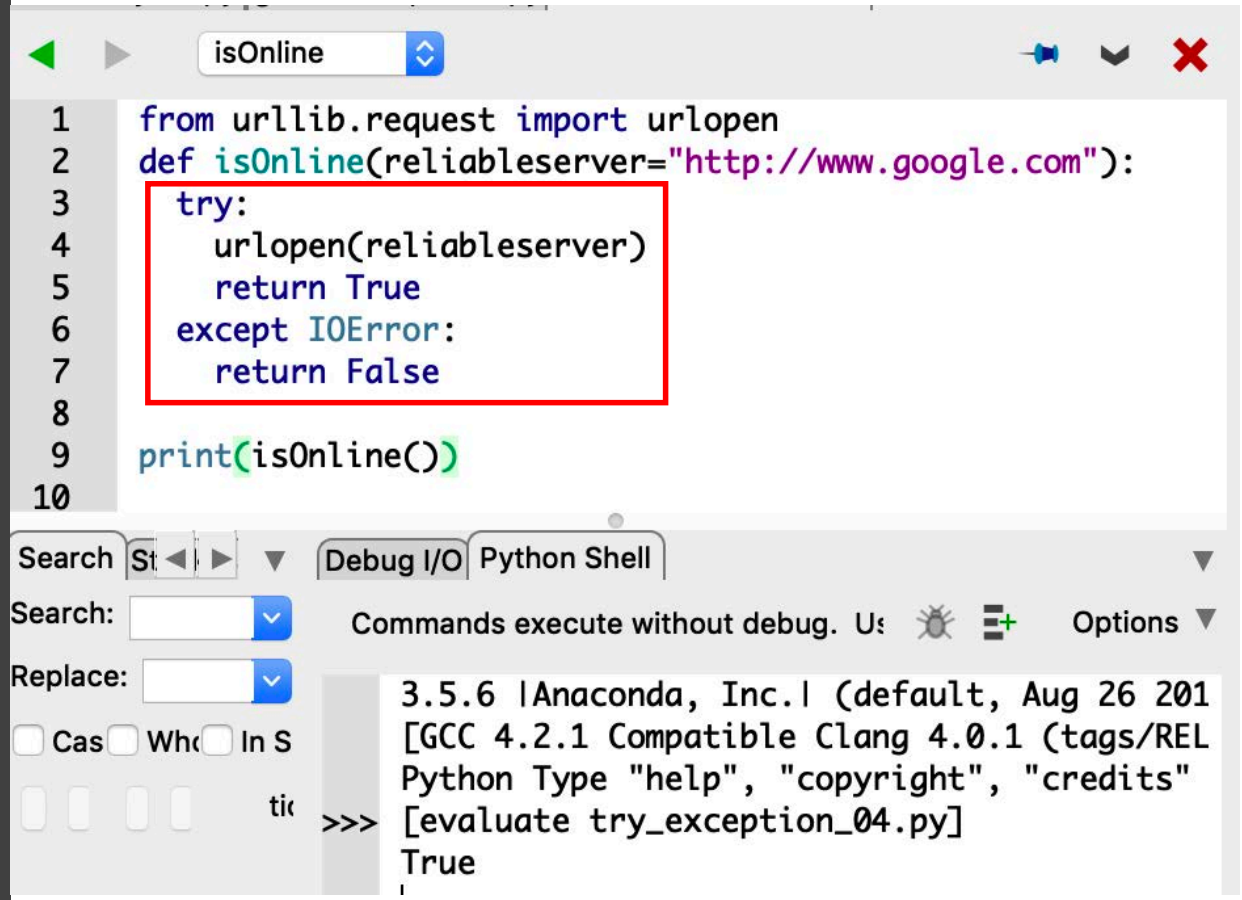
try .. except

- 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 .. except

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



String functions

- split() - returns a list of strings after breaking the given string by the specified separator.

- Join() - returns a string in which the string elements of sequence have been joined by str separator.

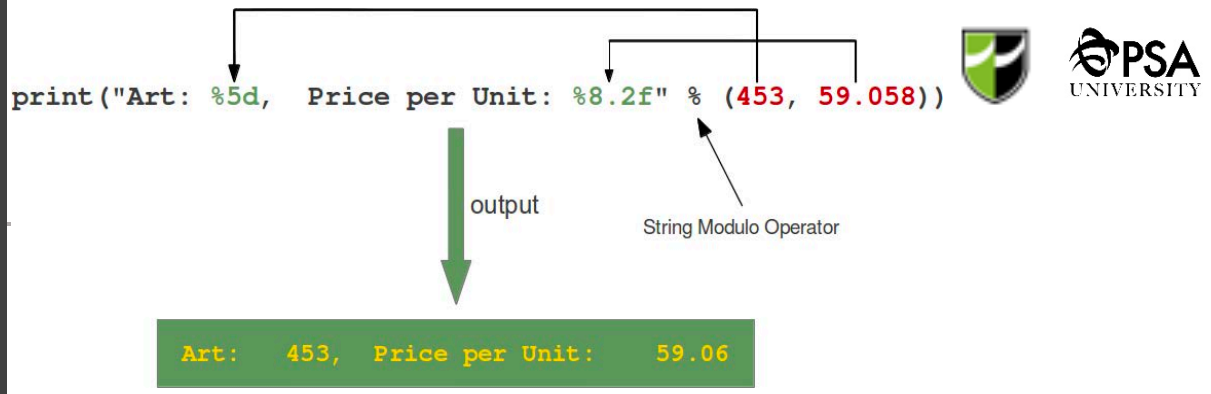
```
1  a = "python or java"
2  b = a.split(" ")
3  print(b)
4  # ['python', 'or', 'java']
5
6  b = a.split("on")
7  print(b)
8  # ['pyth', ' or java']
9
10 a_list = ['python', 'or', 'java']
11 seperator = " "
12 b = seperator.join(a_list)
13 print(b)
14 # "python or java"
15
16 seperator = ","
17 c = seperator.join(a_list)
18 print(c)
19 # "python,or,java"
```

```
Search
Search:
Replace:
Debug I/O  Python Shell
Commands execute without debug  Options
3.5.6 |Anaconda, Inc.| (default, Aug 26
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/
Python Type "help", "copyright", "credit
>>> [evaluate string_function_01.py]
['python', 'or', 'java']
['pyth', ' or java']
python or java
python,or,java
>>>
```

Exercise
Find Longest Word

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

String formatting

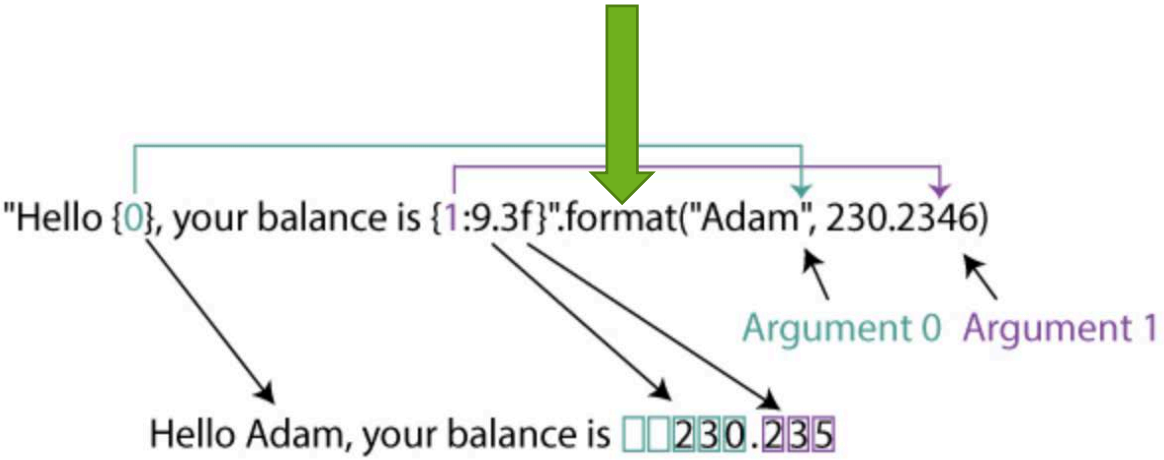



```python
import math
a = math.pi
print(a)
# 3.141592653589793

b = 5
c = "python"
line = "%s %f %d" % (c, a, b)
print(line)
# python 3.141593 5

line = "%03d" % (b)
print(line)
# 005
```

```
Search | Debug I/O | Python Shell
Search:
Replace:
Commands execute without debug. Options
3.5.6 |Anaconda, Inc.| (default, Aug 26 201
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/REL
Python Type "help", "copyright", "credits"
>>> [evaluate string_fromat_01.py]
3.141592653589793
python 3.141593 5
005
```

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

String formatting



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

Exercise - string formatting

Given the variable
I = “admin:$E*G$@R:/users/root:”

Can you print it like
User : admin
Password : $E*G$@R
Homedir : /users/root

Exercise – Xmas Tree

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

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

The Random. Library

implements pseudo-random number generators for various distributions.

```
1  import random
2
3  # random.randint(a,b)
4  # return a randmon integer N such that a <= N <= b
5  print(random.randint(1,10))
6
7  # random.random()
8  # Return the next random floating point number
9  # in the range [0.0, 1.0]
10 print(random.random())
11
12 # shuffle a list
13 my_list=[1,2,3,4,5]
14 random.shuffle(my_list)
15 print(my_list)
16
17 # randomly chose from a list
18 print(random.choice(my_list))
19
```

```
Search  St
Search:
Replace:
Cas  Wh  In S
ti

Debug I/O  Python Shell
Commands execute without debug.  Us  Options
3.5.6 |Anaconda, Inc.| (default, Aug 26 201
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/REL
Python Type "help", "copyright", "credits"
>>> [evaluate random_01.py]
9
0.9341616466170629
[2, 4, 1, 3, 5]
3
```

Additional reading:
https://docs.python.org/3/library/random.html

The random library

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
```

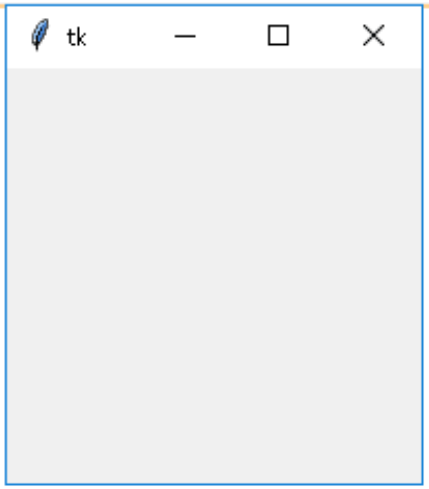
```
What is your name?
John
Well, John, I am thinking of a number between 1 and 20
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your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

Process finished with exit code 0
```

Graphical User Interface

https://wiki.python.org/moin/GuiProgramming

Tkinter – Python's standard GUI library
It is a commonly used GuiProgramming toolkit for Python.

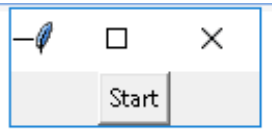
```
1 import tkinter
2
3 window = tkinter.Tk()
4 window.mainloop()
5
```



Graphical User Interface

Add a button

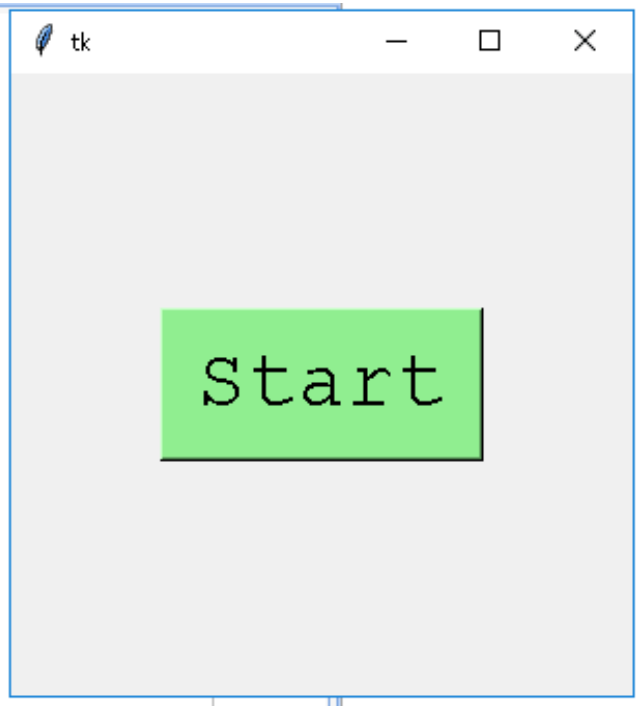
```
1  import tkinter
2
3  window = tkinter.Tk()
4
5  # Add a button
6  button1 = tkinter.Button(window, text="Start")
7  button1.pack()
8
9  window.mainloop()
10
```



Graphical User Interface

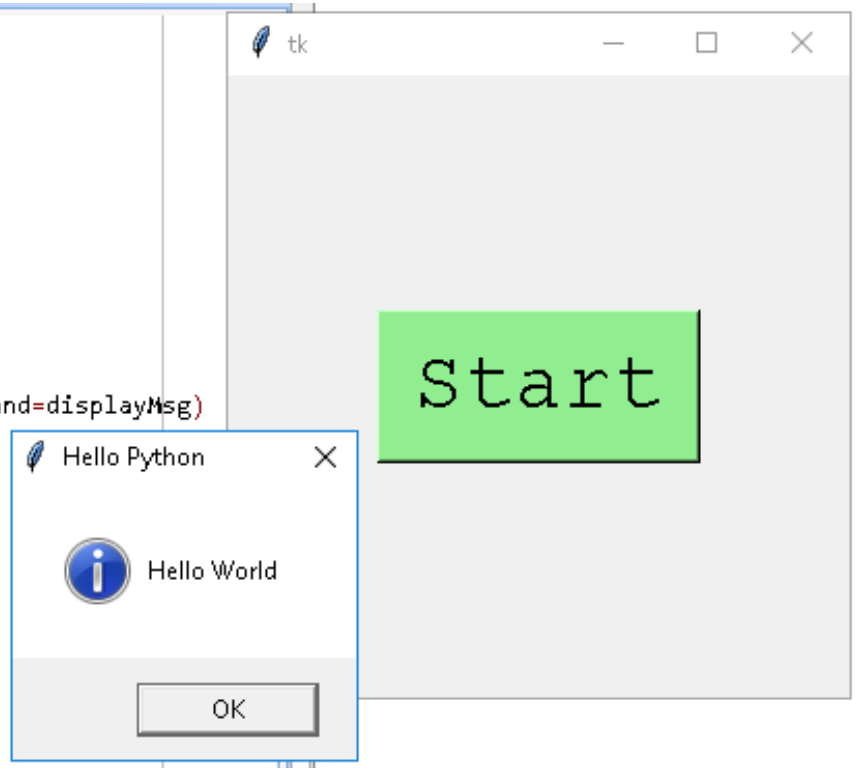
Set the window's size.
Configure the colour and position of the button.

```
1  import tkinter
2  import tkinter.messagebox
3
4  window = tkinter.Tk()
5
6  # Set the window's size
7  window.geometry("300x300")
8
9  # Add and configure a button
10 button1 = tkinter.Button(window, text="Start", bg="lightgreen")
11 button1.config(font=("Courier",30))
12 button1.pack(side="top", expand=tkinter.YES)
13
14 window.mainloop()
15
```



Graphical User Interface

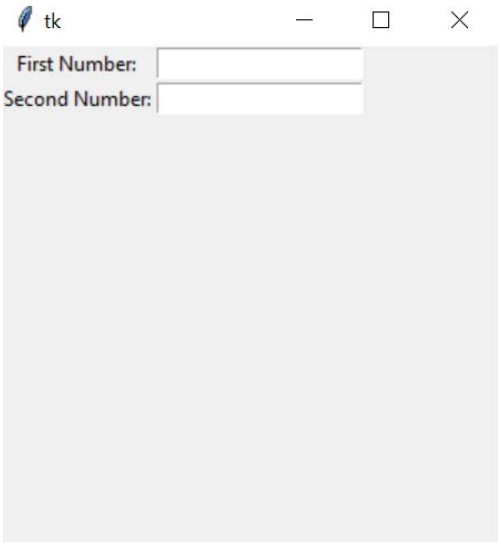
```
1  import tkinter
2  import tkinter.messagebox
3
4  window = tkinter.Tk()
5
6  # Set the window's size
7  window.geometry("300x300")
8
9  def displayMsg():
10     tkinter.messagebox.showinfo("Hello Python", "Hello World")
11
12 # Add and configure a button
13 button1 = tkinter.Button(window, text="Start", bg="lightgreen", command=displayMsg)
14 button1.config(font=("Courier",30))
15 button1.pack(side="top", expand=tkinter.YES)
16
17 window.mainloop()
18
19
```



Graphical User Interface

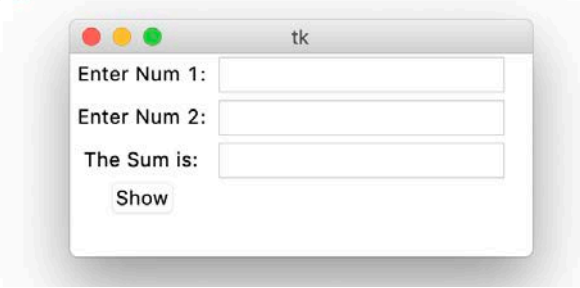
- Specify position and size of UI components
  - Use grid to arrange component in row and column

```
from  tkinter  import  *

master  = Tk()
master .geometry  ("300x300"  )

l1 = Label (master , text ="First Number:")
l2 = Label (master , text ="Second Number:")
l1.grid (row =0, column =0)
l2.grid (row =1, column  =0)

e1  = Entry (master )
e2  = Entry (master )
e1.grid (row =0, column =1)
e2.grid (row =1, column =1)

mainloop  ( )
```



Entry Widget

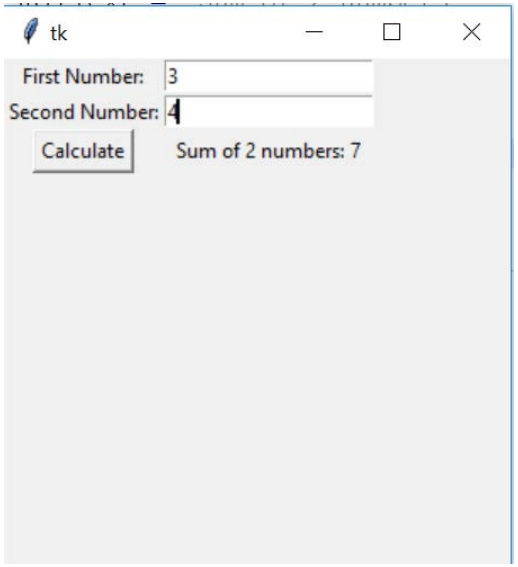
```
1  import tkinter
2  import tkinter.messagebox
3
4  window = tkinter.Tk()
5
6  #set the window's size
7  window.geometry("300x300")
8
9  def show_answer():
10     Ans = int(num1.get()) + int(num2.get())
11     print(Ans)
12     ans.insert(0,Ans)
13
14 label1 = tkinter.Label(window, text = "Enter Num 1:").grid(row=0)
15 label2 = tkinter.Label(window, text = "Enter Num 2:").grid(row=1)
16 label3 = tkinter.Label(window, text = "The Sum is:").grid(row=2)
17
18 num1 = tkinter.Entry(window)
19 num2 = tkinter.Entry(window)
20 ans = tkinter.Entry(window)
21
22 num1.grid(row=0, column=1)
23 num2.grid(row=1, column=1)
24 ans.grid(row=2, column=1)
25
26 #Add a button
27 button1 = tkinter.Button(window, text="Show", bg="gray", command=show_answer)
28 button1.grid(row=4, column=0)
29
30 window.mainloop()
31
32
```



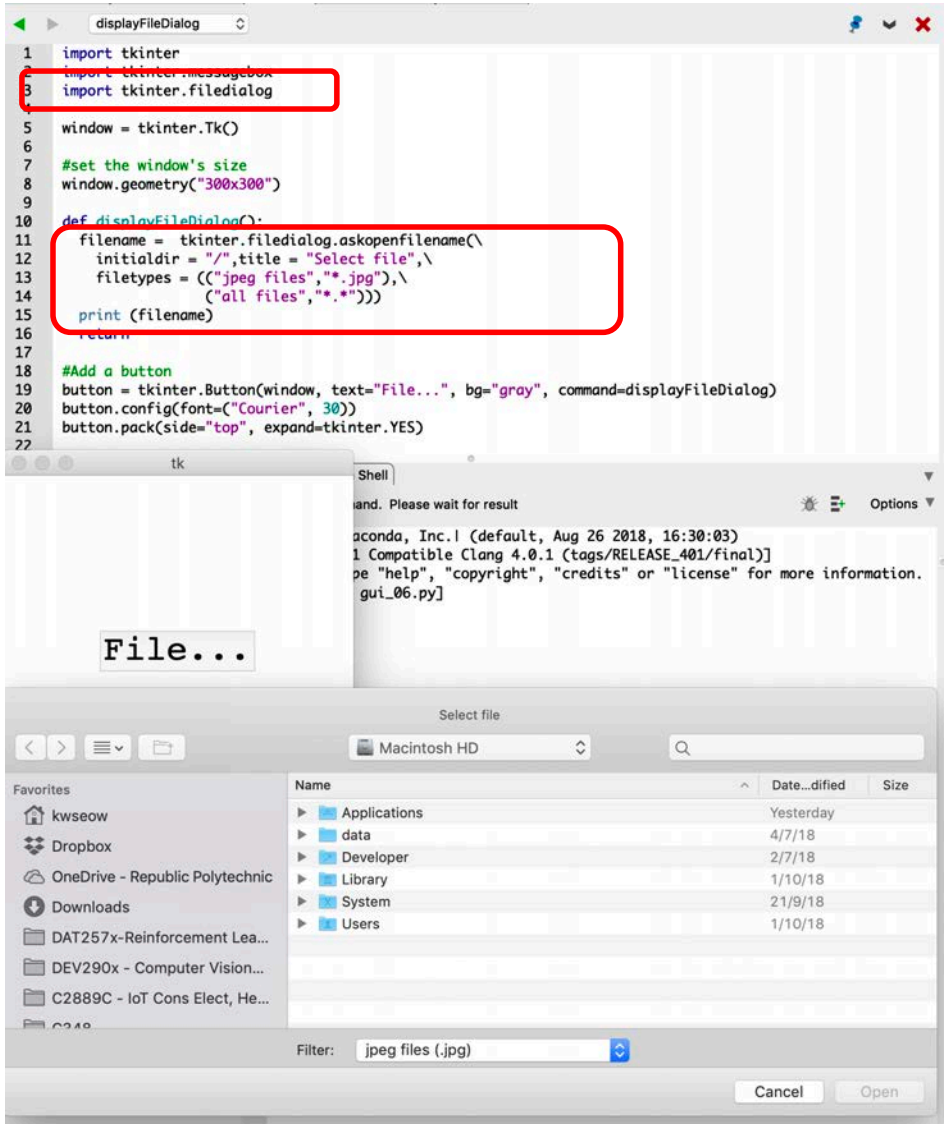
Graphical User Interface

- Getting input, and display result in label

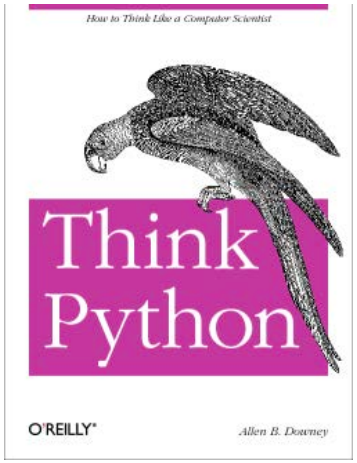
```
from tkinter import *

def calculate():
    total = int(e1.get()) + int(e2.get())
    resultText = "Sum of 2 numbers: " + str(total)
    resultLabel.config(text=resultText)

master = Tk()
master.geometry("300x300")

l1 = Label(master, text="First Number:")
l2 = Label(master, text="Second Number:")
l1.grid(row=0, column=0)
l2.grid(row=1, column =0)

e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

Button(master, text='Calculate',command=calculate).grid(row=2, column=0)
resultLabel = Label(master, text="Answer: ")
resultLabel.grid(row=2, column=1)

mainloop( )
```



Filedialog

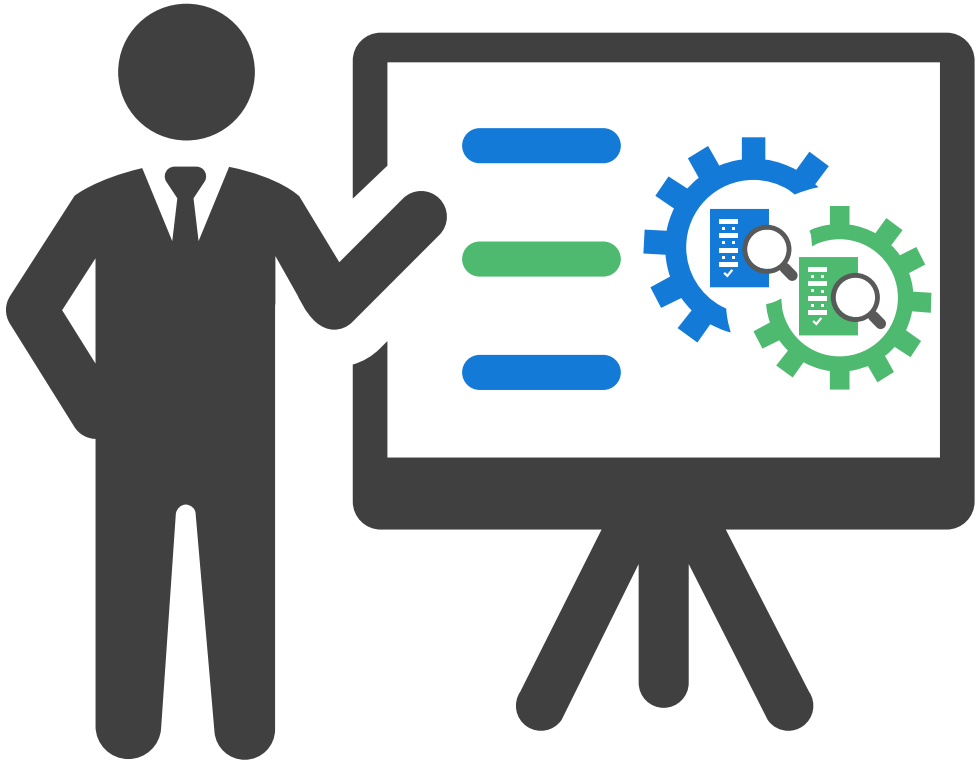

displayFileDialog
import tkinter
import tkinter.filedialog
window = tkinter.Tk()
#set the window's size
window.geometry("300x300")
def displayFileDialog():
filename = tkinter.filedialog.askopenfilename(\
initialdir = "/",title = "Select file",\
filetypes = (("jpeg files","*.jpg"),\
("all files","*.*")))
print (filename)
#Add a button
button = tkinter.Button(window, text="File...", bg="gray", command=displayFileDialog)
button.config(font=("Courier", 30))
button.pack(side="top", expand=tkinter.YES)
tk
File...
Shell
Options
Select file
Macintosh HD
Favorites
kwseow
Dropbox
OneDrive - Republic Polytechnic
Downloads
DAT257x-Reinforcement Lea...
DEV290x - Computer Vision...
C2889C - IoT Cons Elect, He...
Name
Date...dified
Size
Applications
Yesterday
data
4/7/18
Developer
2/7/18
Library
1/10/18
System
21/9/18
Users
1/10/18
Filter:
jpeg files (.jpg)
Cancel
Open

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.

Day 1 Summary



- ✓ *A brief history of Python*
- ✓ *Setting up python environment*
- ✓ *Learn the basics:*
  - ✓ *Data types*
  - ✓ *Conversions*
  - ✓ *String operations*
  - ✓ *Functions*
  - ✓ *And more!*
- ✓ *Try/except*
- ✓ *String functions, formatting*
- ✓ *Graphical User Interface*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: http://bit.ly/2XI57hB

Thank you