Python for Data Science

Welcome!

Welcome to the Python for Data Science class!

About the Program

Four Part Series, Four Hours Each

- Session 1: Getting Started with Python
- Session 2: Applying Python – The Basics
- Session 3: Exploring Python Files, Dictionaries, Sets & Functions
- Session 4: Expanding Python – Functions, Error Handling, Importing and OO Classes

Quick Logistics: Format, Q&A and Follow-On Materials / Hand-Outs

About Me: Ernesto Lee

Today's Agenda: Session 1

Get started with Python!

Leave with an understanding of how to create a sandbox environment so that you can practice Python at any time!

- How to get started with Python
- Tools and environments for writing Python code
- How and Why we test software

Topics We'll Explore Today:

An Overview of Python

- What is python?
- Python Timeline
- Advantages/Disadvantages of Python
- Getting help with pydoc

The Python Environment

- Starting Python
- Using the interpreter
- Running a Python script
- Python scripts on Unix/Windows
- Editors and IDEs

Getting Started

- Using variables
- Builtin functions
- Strings
- Numbers
- Converting among types
- Writing to the screen
- Command line parameters

Python Overview

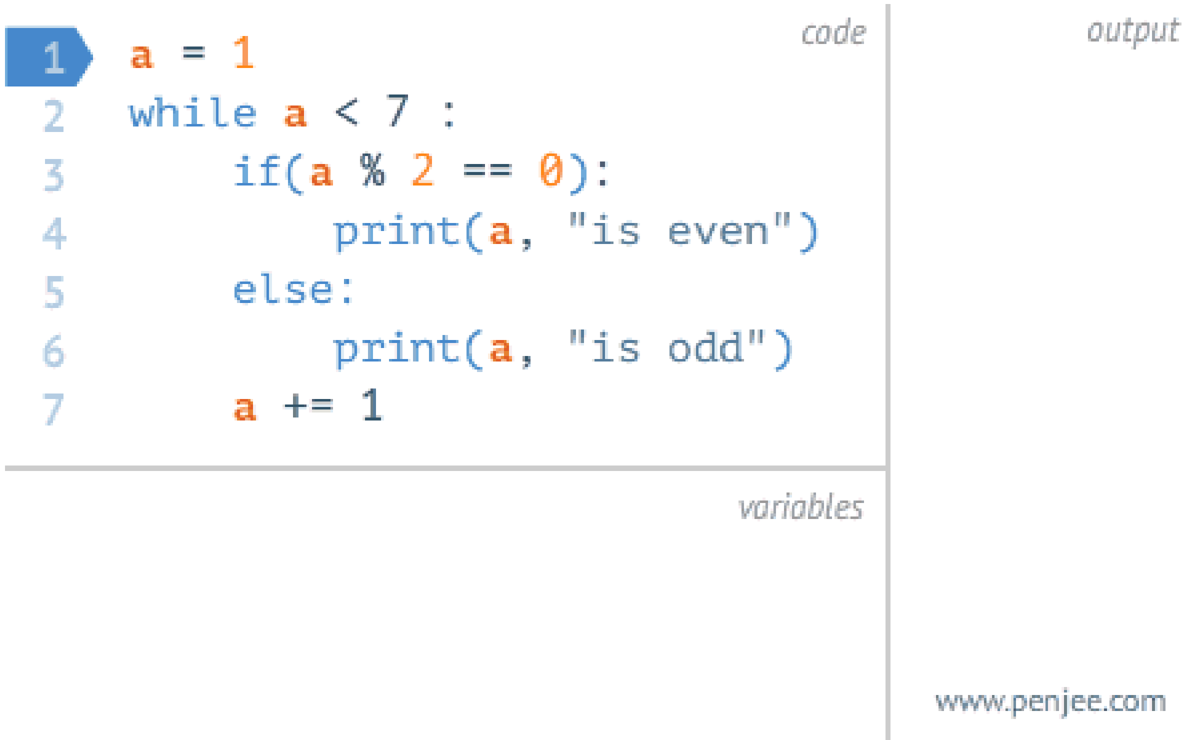
About Python

This lesson covers

- Why and what is Python?
- What Python does really well
- What Python doesn’t do as well
- Why learn Python 3?

Why use Python?

- Choosing a language to learn is difficult.
- Python is a good choice for many programming problems, (especially data analytics)

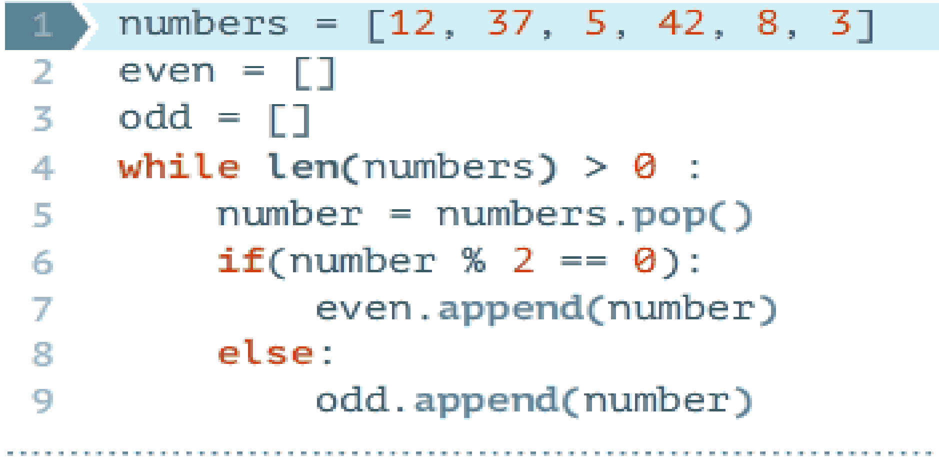


```python
a = 1
while a < 7 :
    if(a % 2 == 0):
        print(a, "is even")
    else:
        print(a, "is odd")
    a += 1
```

GUIDO VAN ROSSUM

Creator of Python

Python is easy to use



```python
numbers = [12, 37, 5, 42, 8, 3]
even = []
odd = []
while len(numbers) > 0 :
    number = numbers.pop()
    if(number % 2 == 0):
        even.append(number)
    else:
        odd.append(number)
```

www.penjee.com

Programmers familiar with traditional languages will find it easy to learn Python. All of the familiar constructs—loops, conditional statements, arrays, and so forth—are included, but many are easier to use in Python.

Python is expressive

- To get an idea of how Python's expressiveness can simplify code, consider swapping the values of two variables, var1 and var2.
- In a language like Java, this requires three lines of code and an extra variable:

```
int temp = var1;
var1 = var2;
var2 = temp;
```

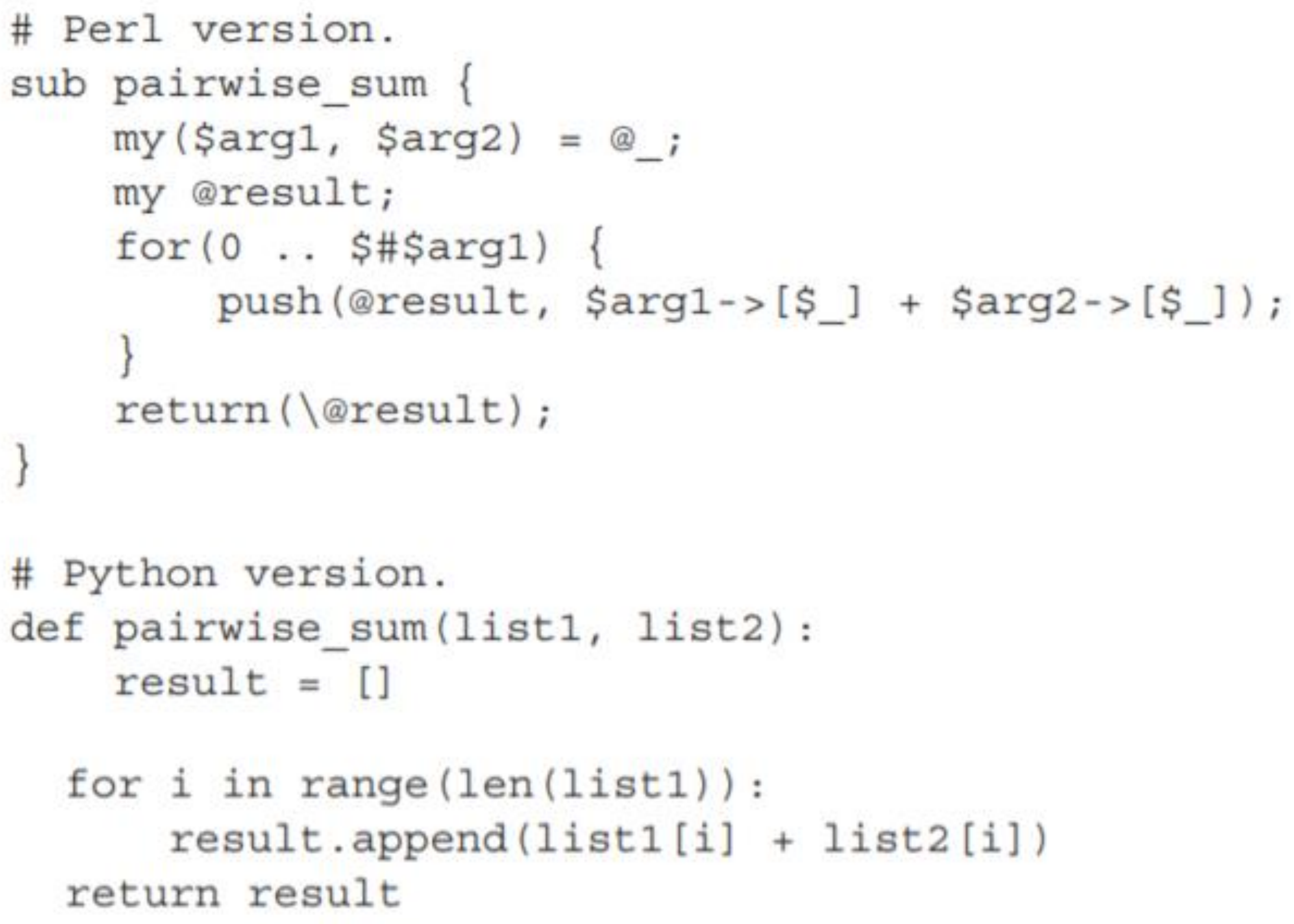
Python is expressive

- Python lets you make the same swap in one line and in a way that makes it obvious that a swap of values has occurred:

```
var2, var1 = var1, var2
```

- Of course, this is a very simple example, but you find the same advantages throughout the language.

Python is readable



```
# Perl version.
sub pairwise_sum {
    my($arg1, $arg2) = @_;
    my @result;
    for(0 .. $#$arg1) {
        push(@result, $arg1->[$_] + $arg2->[$_]);
    }
    return(\@result);
}

# Python version.
def pairwise_sum(list1, list2):
    result = []

  for i in range(len(list1)):
      result.append(list1[i] + list2[i])
  return result
```

Python is complete—“batteries included”

- For example, with Python, you can write a web server to share the files in a directory with just two lines of code:

```
import http.server
http.server.test(HandlerClass=http.server.SimpleHTTPRequestHandler)
```

- There’s no need to install libraries to handle network connections and HTTP; it’s already in Python, right out of the box

Python is cross-platform

- Python is also an excellent cross-platform language. Python runs on many platforms: Windows, Mac, Linux, UNIX, and so on.

- Because it’s interpreted, the same code can run on any platform that has a Python interpreter, and almost all current platforms have one.

Python is free

- Python is free.
- Python was originally, and continues to be, developed under the open source model, and it’s freely available.

What Python doesn’t do as well

- Although Python has many advantages, no language can d everything, so Python isn’t the perfect solution for all your needs.

Python isn’t the fastest language

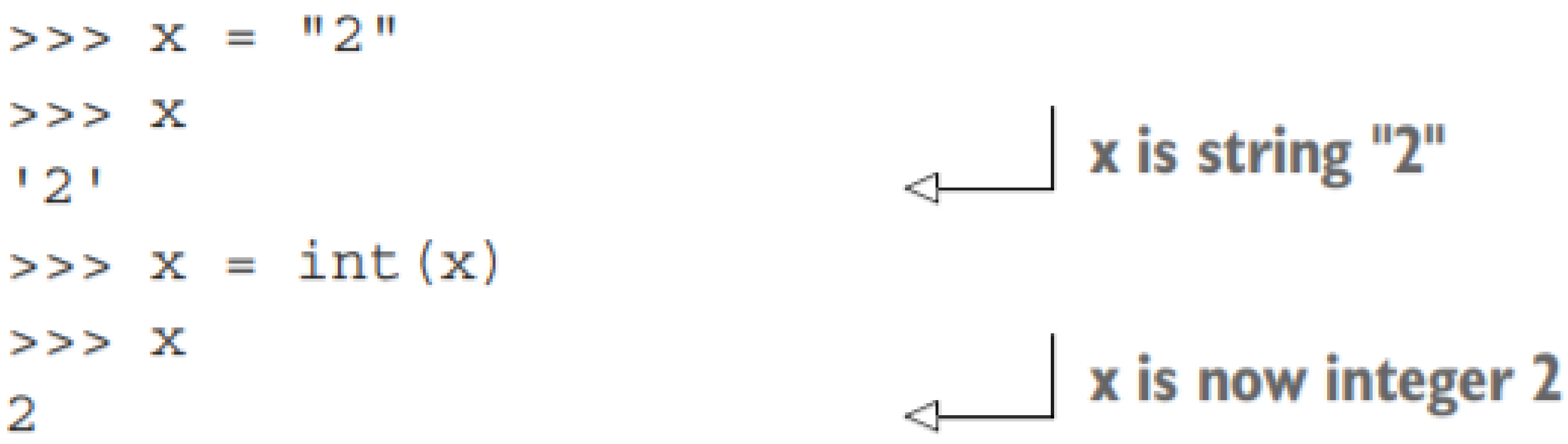
- A possible drawback with Python is its speed of execution.
- It isn’t a fully compiled language. Instead, it’s first compiled to an internal bytecode form, which is then executed by a Python interpreter.

Python doesn’t have the most libraries

- Although Python comes with an excellent collection of libraries, and many more are available, Python doesn’t hold the lead in this department.
- Languages like C, Java, and Perl have even larger collections of libraries available

Python doesn’t check variable types at compile time

- It’s possible to use the variable x to refer to a string in one line and an integer in another:



```
>>> x = "2"
>>> x
'2'
>>> x = int(x)
>>> x
2
```

x is string "2"

x is now integer 2

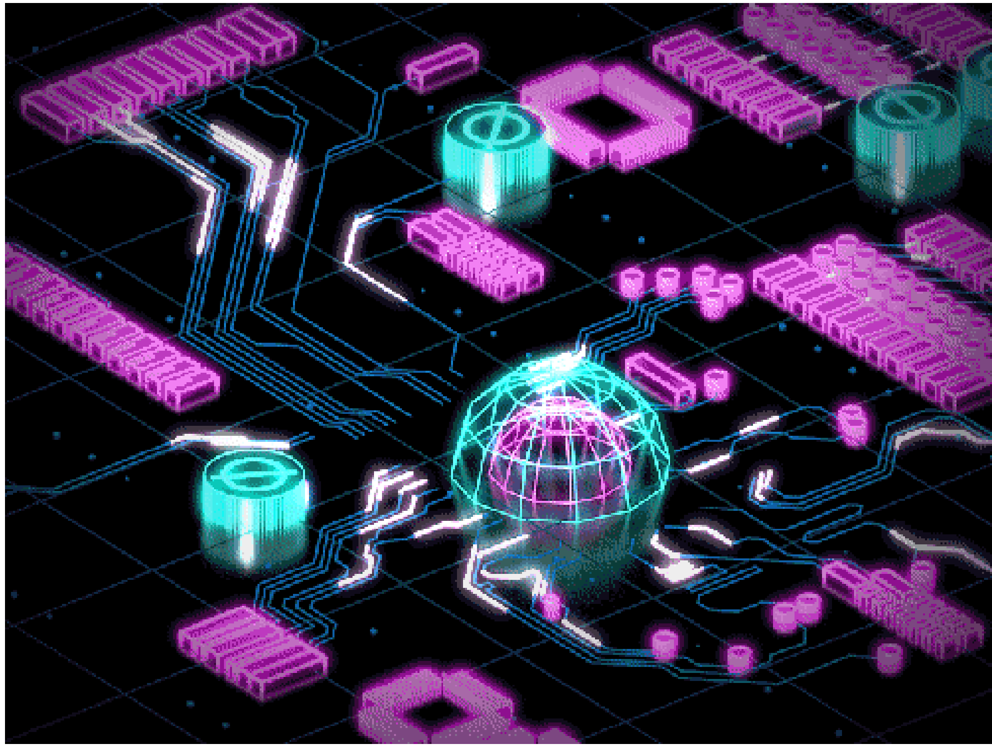
Python doesn’t have as much mobile support

- In the past decade the numbers and types of mobile devices have exploded, and smartphones, tablets, phablets, Chrome courses, and more are everywhere, running on a variety of operating systems.

- Python isn’t as strong player in this space.

Python doesn’t use multiple processors well

- The standard implementation of Python isn’t designed to use multiple cores, due to a feature called the global interpreter lock (GIL).



Why learn Python 3?

- In earlier versions of Python, for example, the print statement didn’t require parentheses around its arguments:

```
print "hello“
```

- In Python 3, print is a function and needs the parentheses:

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

Quick Review

- Python is a modern, high-level language with dynamic typing and simple, consistent syntax and semantics.

- Python is multiplatform, highly modular, and suited for both rapid development and large-scale programming.

The Python Environment

Getting started

This lesson covers

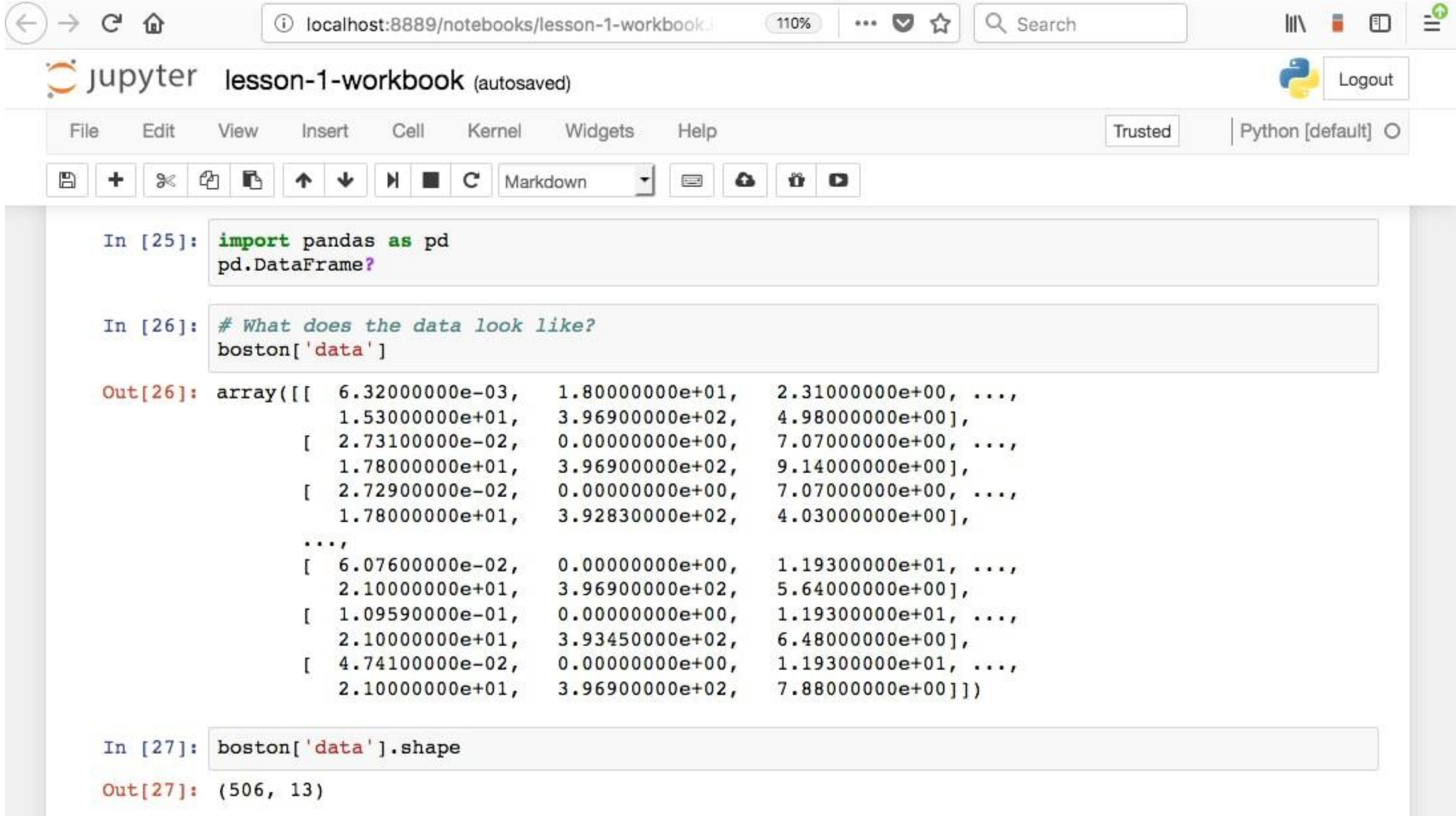
- Starting Python
- Using the interpreter
- Running a Python script
- Python scripts on Unix/Windows
- Editors and IDEs

Installing Python

- Installing Python is a simple matter, regardless of which platform you’re using.
- You’ll need Python v3.6 or better for this course.
- To obtain the latest approved version, please reach out to your Learning Support Team.

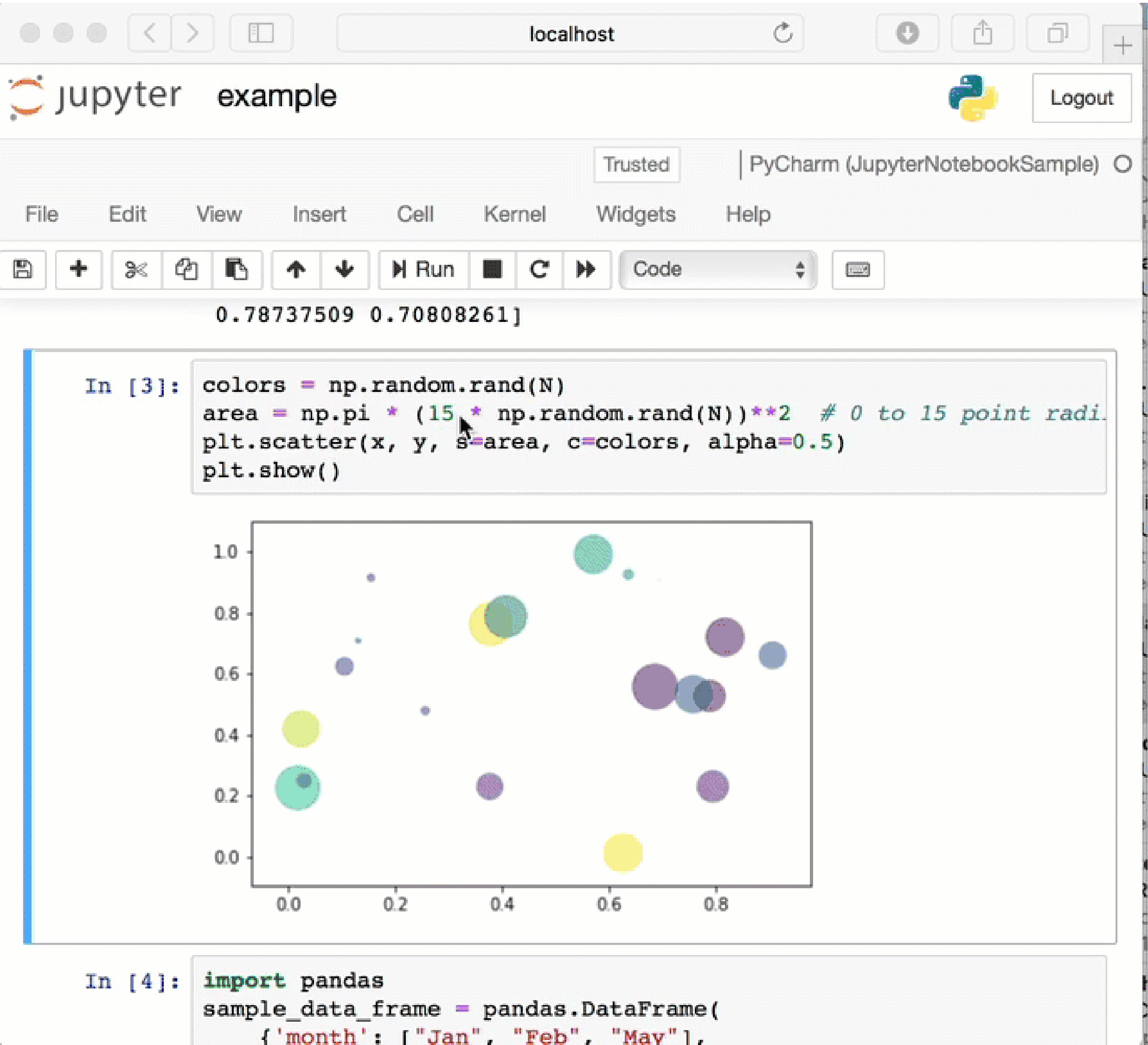
Jupyter Notebooks

- You have many options for accessing interactive Python and one of the more popular IDEs is: Jupyter.



```
In [25]: import pandas as pd
         pd.DataFrame?

In [26]: # What does the data look like?
         boston['data']

Out[26]: array([[  6.32000000e-03,   1.80000000e+01,   2.31000000e+00, ...,
                   1.53000000e+01,   3.96900000e+02,   4.98000000e+00],
                [  2.73100000e-02,   0.00000000e+00,   7.07000000e+00, ...,
                   1.78000000e+01,   3.96900000e+02,   9.14000000e+00],
                [  2.72900000e-02,   0.00000000e+00,   7.07000000e+00, ...,
                   1.78000000e+01,   3.92830000e+02,   4.03000000e+00],
                ...,
                [  6.07600000e-02,   0.00000000e+00,   1.19300000e+01, ...,
                   2.10000000e+01,   3.96900000e+02,   5.64000000e+00],
                [  1.09590000e-01,   0.00000000e+00,   1.19300000e+01, ...,
                   2.10000000e+01,   3.93450000e+02,   6.48000000e+00],
                [  4.74100000e-02,   0.00000000e+00,   1.19300000e+01, ...,
                   2.10000000e+01,   3.96900000e+02,   7.88000000e+00]])

In [27]: boston['data'].shape

Out[27]: (506, 13)
```

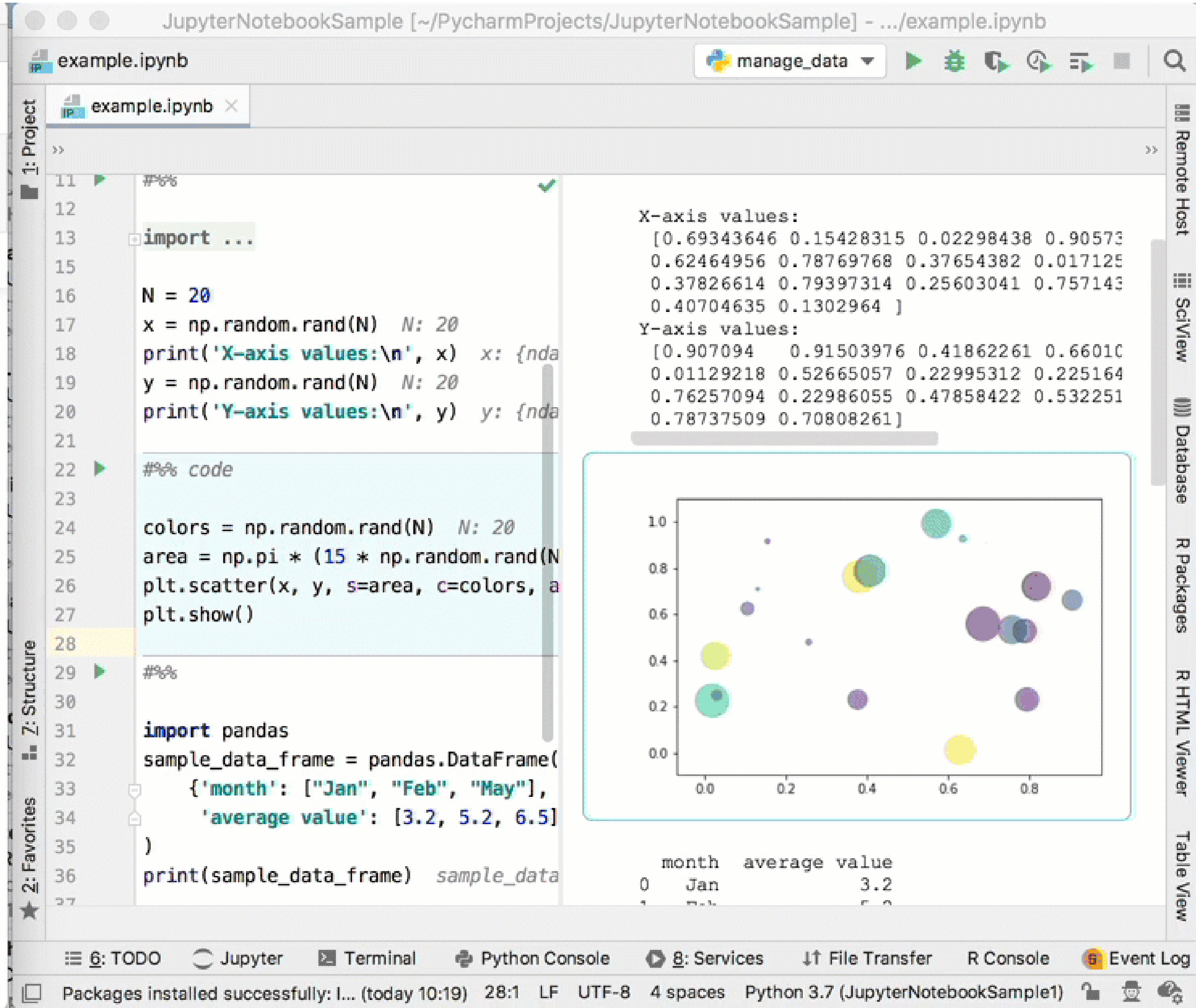
What EXACTLY are Jupyter Notebooks



```
 0.78737509 0.70808261]
```

In [3]:
```
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 0 to 15 point radi
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

In [4]:
```
import pandas
sample_data_frame = pandas.DataFrame(
    {'month': ["Jan", "Feb", "May"],
```

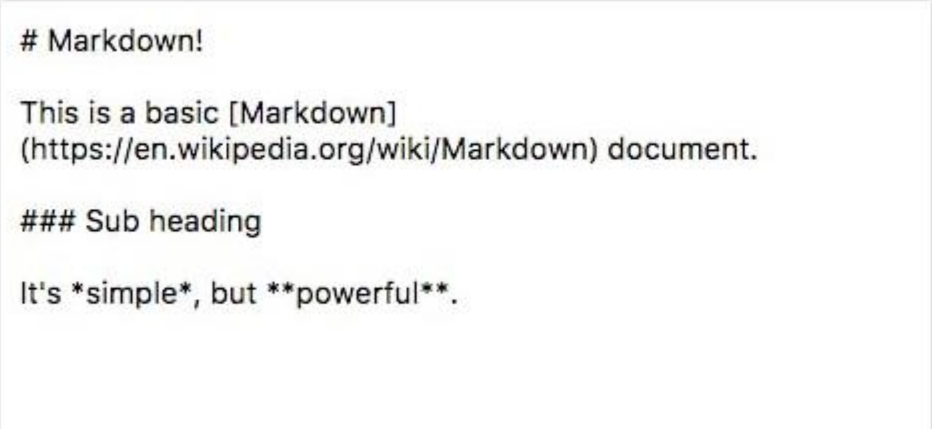


```
#%%

import ...

N = 20
x = np.random.rand(N)  N: 20
print('X-axis values:\n', x)  x: {nda
y = np.random.rand(N)  N: 20
print('Y-axis values:\n', y)  y: {nda

#%% code

colors = np.random.rand(N)  N: 20
area = np.pi * (15 * np.random.rand(N
plt.scatter(x, y, s=area, c=colors, a
plt.show()

#%%

import pandas
sample_data_frame = pandas.DataFrame(
    {'month': ["Jan", "Feb", "May"],
     'average value': [3.2, 5.2, 6.5]
)
print(sample_data_frame)  sample_data
```

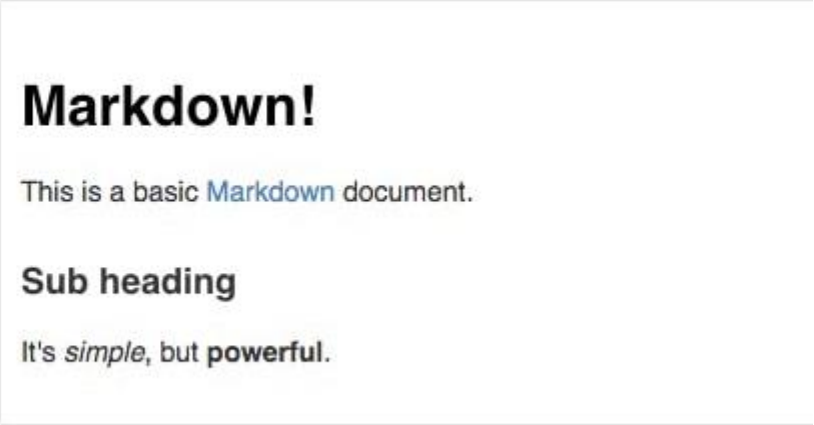
```
X-axis values:
 [0.69343646 0.15428315 0.02298438 0.90573
 0.62464956 0.78769768 0.37654382 0.017125
 0.37826614 0.79397314 0.25603041 0.757143
 0.40704635 0.1302964 ]
Y-axis values:
 [0.907094   0.91503976 0.41862261 0.66010
 0.01129218 0.52665057 0.22995312 0.225164
 0.76257094 0.22986055 0.47858422 0.532251
 0.78737509 0.70808261]
```

```
  month  average value
0   Jan            3.2
```

Markdown



```
# Markdown!

This is a basic [Markdown]
(https://en.wikipedia.org/wiki/Markdown) document.

### Sub heading

It's *simple*, but **powerful**.
```

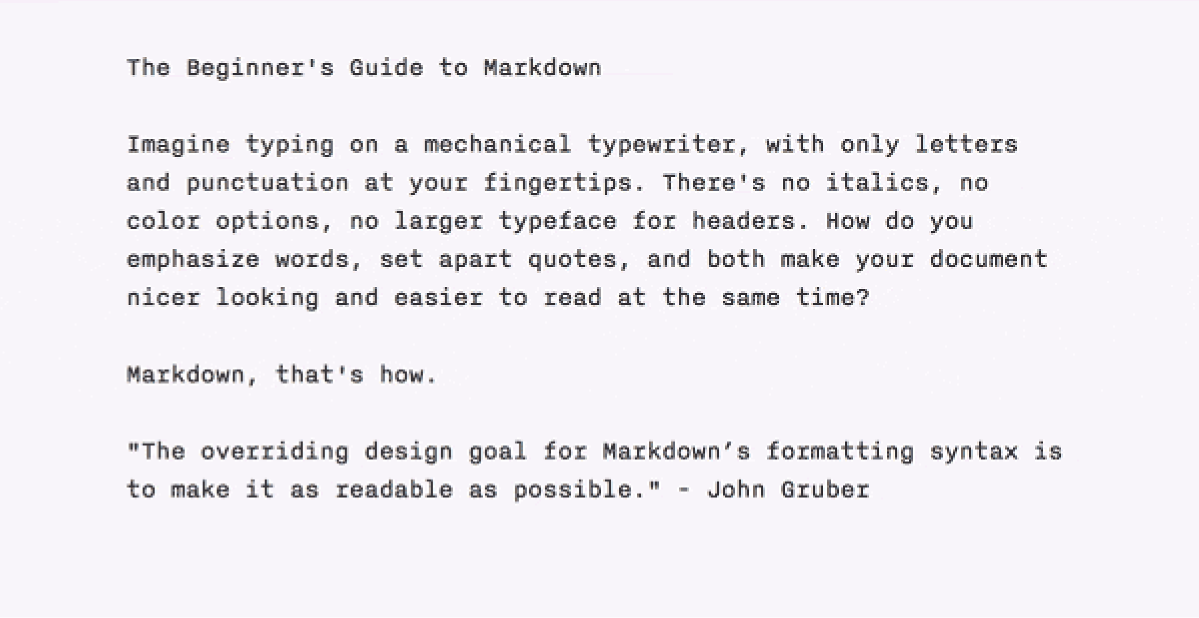


Markdown!

This is a basic Markdown document.

Sub heading

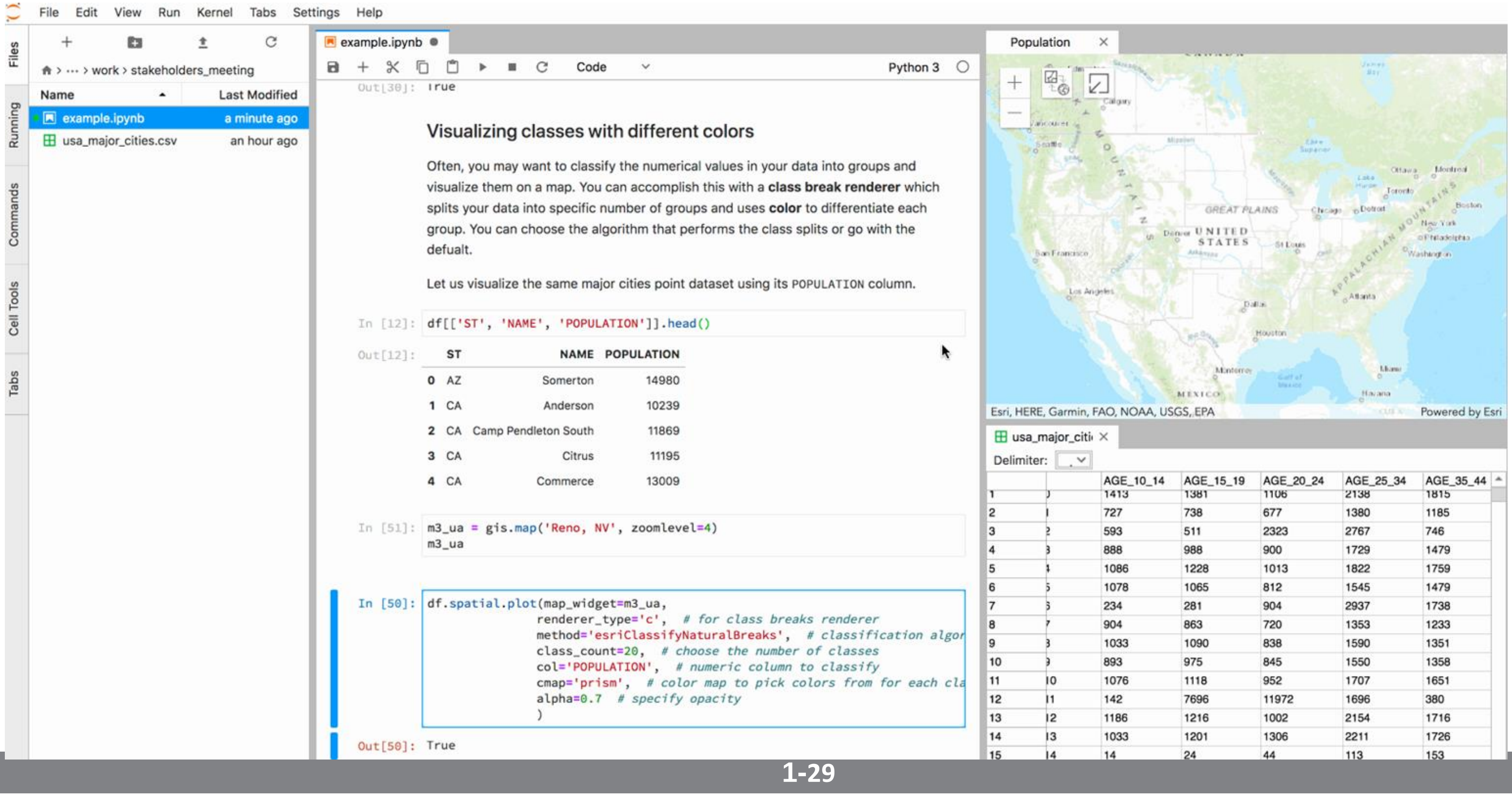
It's *simple*, but **powerful**.



```
The Beginner's Guide to Markdown

Imagine typing on a mechanical typewriter, with only letters
and punctuation at your fingertips. There's no italics, no
color options, no larger typeface for headers. How do you
emphasize words, set apart quotes, and both make your document
nicer looking and easier to read at the same time?

Markdown, that's how.

"The overriding design goal for Markdown's formatting syntax is
to make it as readable as possible." - John Gruber
```

Matter of Perspective

- Lab Style Notebooks
- Deliverable Notebooks



```
Out[30]: True
```

Visualizing classes with different colors

Often, you may want to classify the numerical values in your data into groups and visualize them on a map. You can accomplish this with a **class break renderer** which splits your data into specific number of groups and uses **color** to differentiate each group. You can choose the algorithm that performs the class splits or go with the defualt.

Let us visualize the same major cities point dataset using its `POPULATION` column.

```
In [12]: df[['ST', 'NAME', 'POPULATION']].head()
```

Out[12]:

	ST	NAME	POPULATION
0	AZ	Somerton	14980
1	CA	Anderson	10239
2	CA	Camp Pendleton South	11869
3	CA	Citrus	11195
4	CA	Commerce	13009

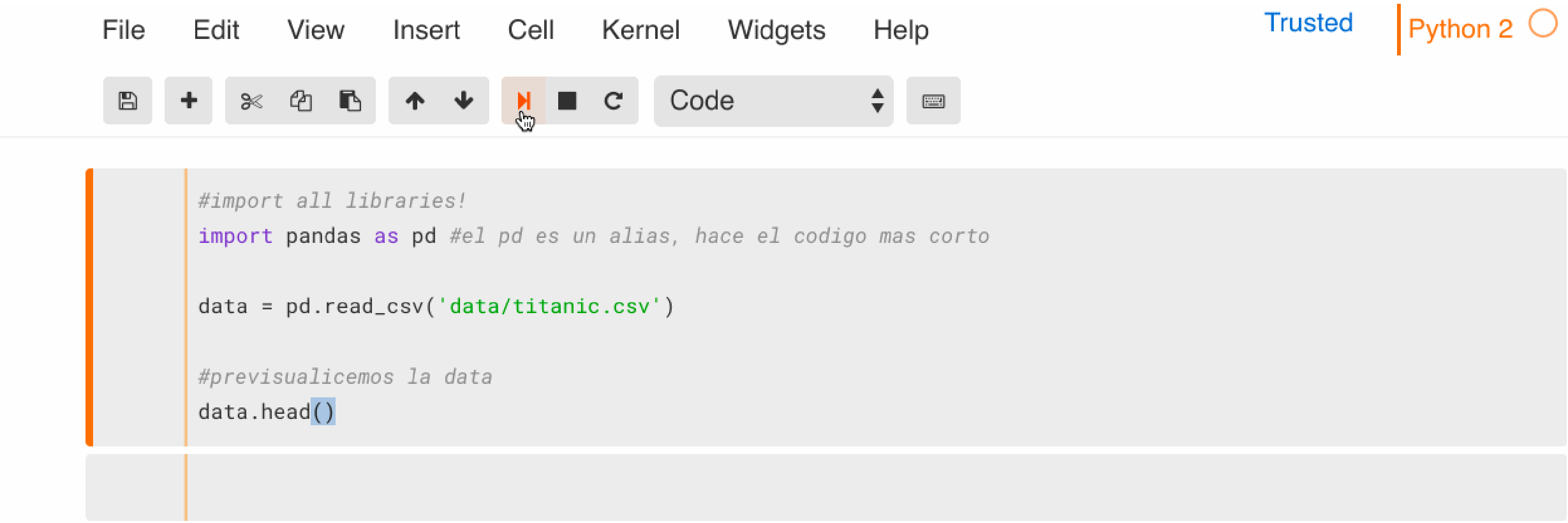
```
In [51]: m3_ua = gis.map('Reno, NV', zoomlevel=4)
         m3_ua
```

```
In [50]: df.spatial.plot(map_widget=m3_ua,
                         renderer_type='c',  # for class breaks renderer
                         method='esriClassifyNaturalBreaks',  # classification algor
                         class_count=20,  # choose the number of classes
                         col='POPULATION',  # numeric column to classify
                         cmap='prism',  # color map to pick colors from for each cla
                         alpha=0.7  # specify opacity
                         )

Out[50]: True
```

		AGE_10_14	AGE_15_19	AGE_20_24	AGE_25_34	AGE_35_44
1	0	1413	1381	1106	2138	1815
2	1	727	738	677	1380	1185
3	2	593	511	2323	2767	746
4	3	888	988	900	1729	1479
5	4	1086	1228	1013	1822	1759
6	5	1078	1065	812	1545	1479
7	6	234	281	904	2937	1738
8	7	904	863	720	1353	1233
9	8	1033	1090	838	1590	1351
10	9	893	975	845	1550	1358
11	10	1076	1118	952	1707	1651
12	11	142	7696	11972	1696	380
13	12	1186	1216	1002	2154	1716
14	13	1033	1201	1306	2211	1726
15	14	14	24	44	113	153

Jupyter Extensibility (through Python Libs)

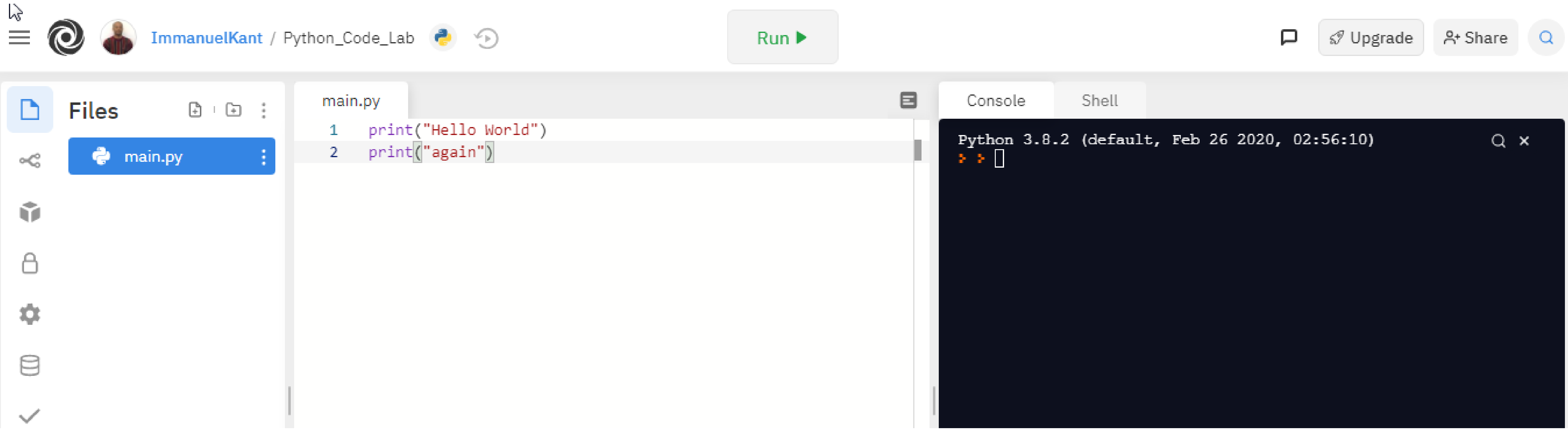
- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn
- Requests
- Bokeh



```
#import all libraries!
import pandas as pd #el pd es un alias, hace el codigo mas corto

data = pd.read_csv('data/titanic.csv')

#previsualicemos la data
data.head()
```

The basic interactive mode

The basic interactive mode is a rather primitive environment, but the interactive examples in this course are generally small. (Jupyter will be the interface of choice)



```
print("Hello World")
print("again")
```

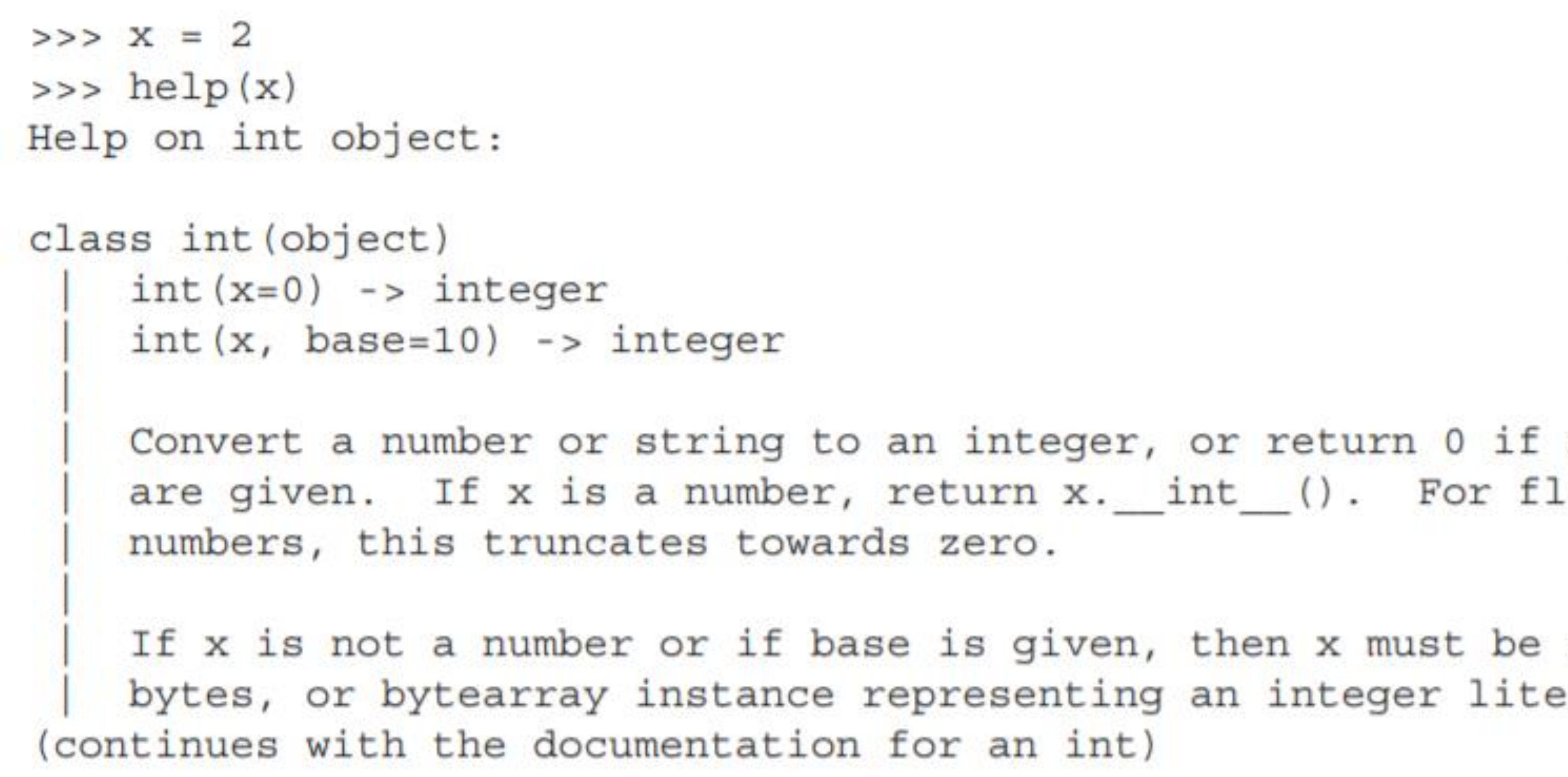
```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
> > 
```

Hello, world

- Use Jupyter or JupyterLab for this (and all exercises and demos)
- Start with the obligatory “Hello, World” program, which is a oneliner in Python (ending each line you type with a hard return):

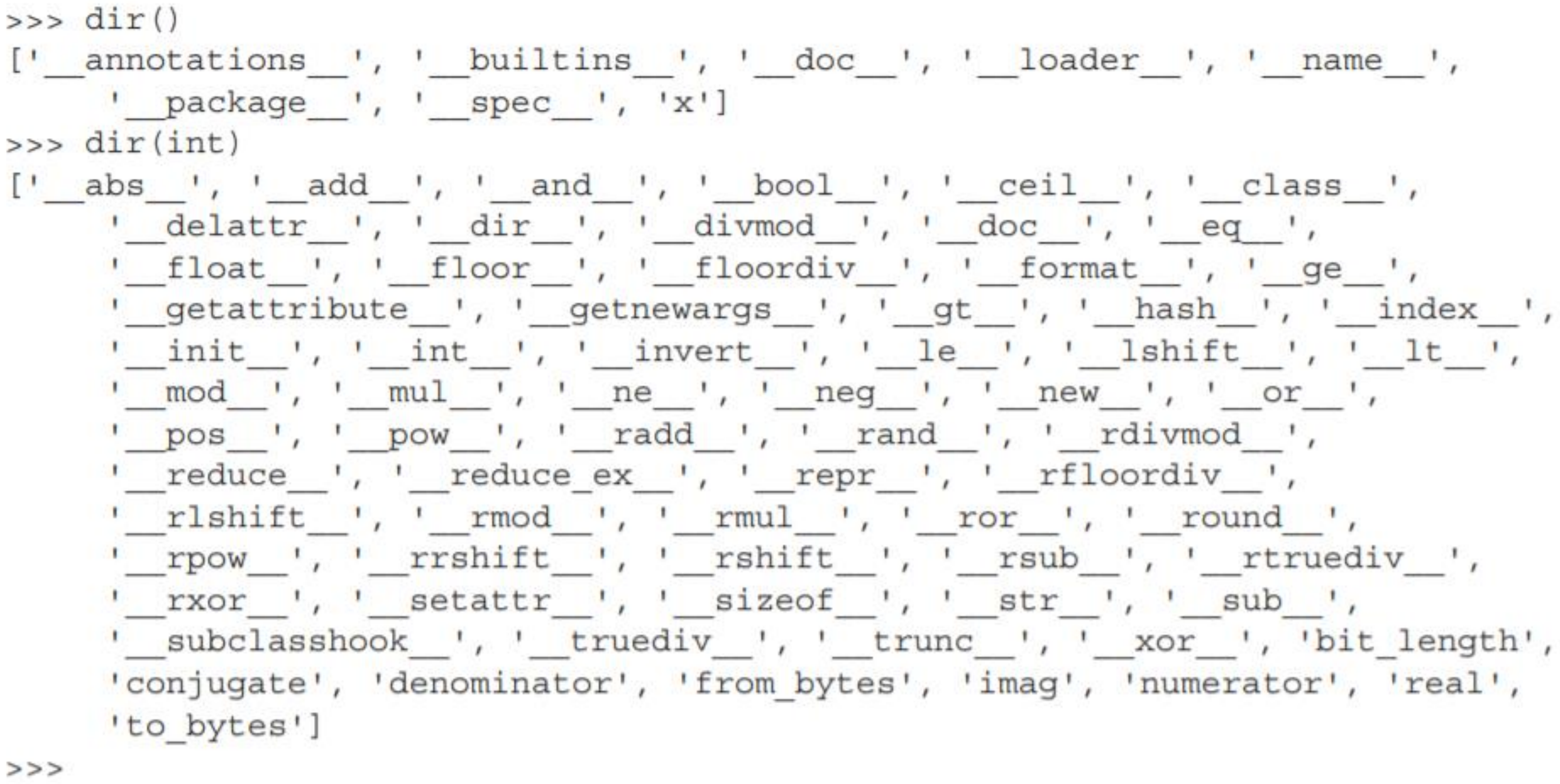
```
>>> print("Hello, World")
Hello, World
```

Using the interactive prompt to explore Python



```
>>> x = 2
>>> help(x)
Help on int object:

class int(object)
 |   int(x=0) -> integer
 |   int(x, base=10) -> integer
 |
 |   Convert a number or string to an integer, or return 0 if no arguments
 |   are given.  If x is a number, return x.__int__().  For floating point
 |   numbers, this truncates towards zero.
 |
 |   If x is not a number or if base is given, then x must be a string,
 |   bytes, or bytearray instance representing an integer literal in the...
(continues with the documentation for an int)
```

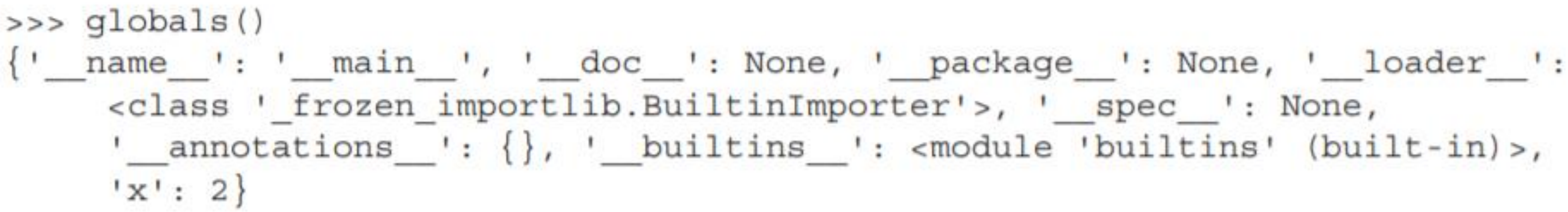
Using the interactive prompt to explore Python



```
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__',
     '__package__', '__spec__', 'x']
>>> dir(int)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
     '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__',
     '__float__', '__floor__', '__floordiv__', '__format__', '__ge__',
     '__getattribute__', '__getnewargs__', '__gt__', '__hash__', '__index__',
     '__init__', '__int__', '__invert__', '__le__', '__lshift__', '__lt__',
     '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__',
     '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__',
     '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__',
     '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__',
     '__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__',
     '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__',
     '__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'bit_length',
     'conjugate', 'denominator', 'from_bytes', 'imag', 'numerator', 'real',
     'to_bytes']
>>>
```

Using the interactive prompt to explore Python

- Unlike dir, both globals and locals show the values associated with the objects.
- In the current situation, both functions return the same thing, so we have only shown the output from globals():



```
>>> globals()
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__':
     <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None,
     '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>,
     'x': 2}
```

The Zen of Python

- In your console – type:

import this

Quick Review

- Installing Python 3 on Windows systems is as simple as downloading the latest installer from www.python.org and running it.
- Installation on Linux, UNIX, and Mac systems will vary
- Refer to installation instructions on the Python website, and use your system’s software package installer where possible.

Getting Started!

The absolute basics

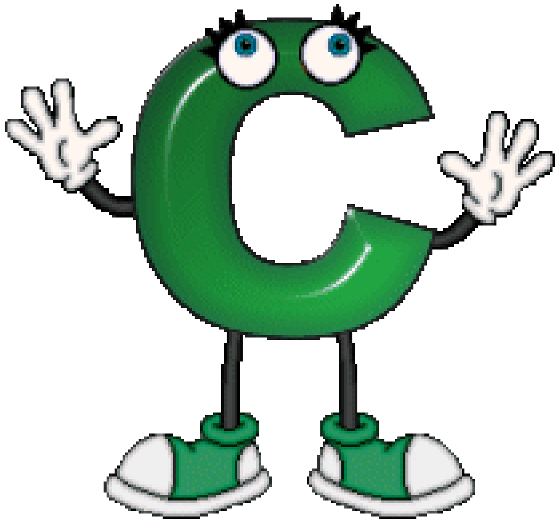
This lesson covers

- Indenting and block structuring
- Differentiating comments
- Assigning variables
- Evaluating expressions
- Using common data types
- Getting user input
- Using correct Pythonic style

Indentation and block structuring

- C code that calculates the factorial of 9, leaving the result in the variable r:

```
/* This is C code */
int n, r;
n = 9;
r = 1;
while (n > 0) {
 r *= n;
 n--;
}
```



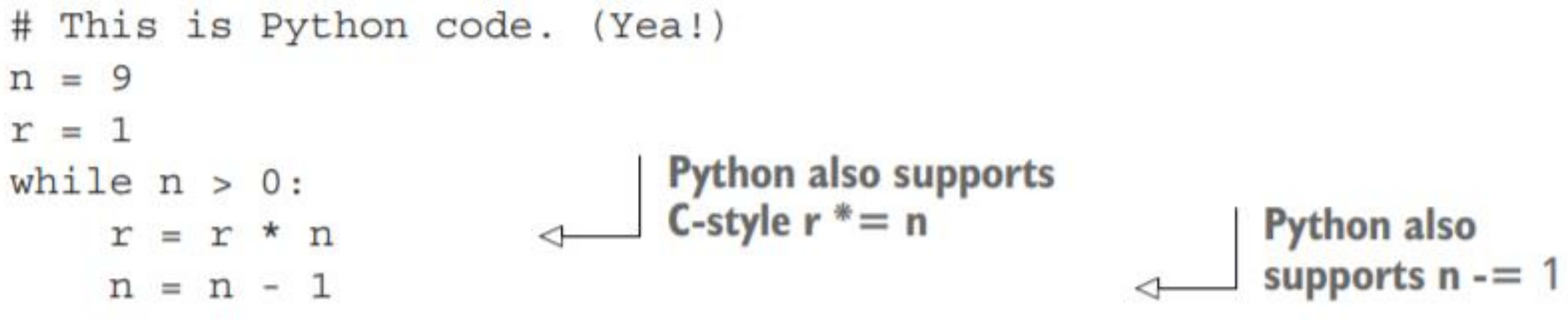
Indentation and block structuring

```
/* And this is C code with arbitrary indentation */
 int n, r;
 n = 9;
 r = 1;
 while (n > 0) {
         r *= n;
         n--;
}
```

Indentation and block structuring

- The code still would execute correctly, even though it’s rather difficult to read.

- Here’s the Python equivalent:



```
# This is Python code. (Yea!)
n = 9
r = 1
while n > 0:
    r = r * n
    n = n - 1
```

Python also supports C-style r *= n

Python also supports n -= 1

Differentiating comments

- For the most part, anything following a # symbol in a Python file is a comment and is disregarded by the language.
- The obvious exception is a # in a string, which is just a character of that string:

```
# Assign 21 to x
x = 21
x = 7 # Now x is 7
x = "# This is not a comment"
```

Variables and assignments

- The most commonly used command in Python is assignment, which looks pretty close to what you might’ve used in other languages.

- Python code to create a variable called x and assign the value 21 to that variable is

```
x = 21
```

Variables and assignments

- Python variables can be set to any object, whereas in C and many other languages, variables can store only the type of value they’re declared as.
- The following is perfectly legal Python code:

```
>>> x = "Hello Ernesto"
>>> print(x)
Hello Ernesto
>>> x = 21
>>> print(x)
21
```

- A new assignment overrides any previous assignments.
- The del statement deletes the variable.

```
>>> x = 21
>>> print(x)
21
>>> del x
>>> print(x)
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>>
```

```
Error

Error: File "Detonation.paf" has crashed.
F4: Reboot         F5: Restart Operation
F6: Rebuild File   F7: Exit
```

Expressions

- Python supports arithmetic and similar expressions; these expressions will be familiar to most readers.

```
x = 13
y = 15
z = (x + y) / 2
```

Strings

- You’ve already seen that Python, like most other programming languages, indicates strings through the use of double quotes.

- This line leaves the string "Hello, World“ in the variable x:

```
x = "Hello, World"
```

Strings

- Backslashes can be used to escape characters, to give them special meanings.

    \n means the newline character

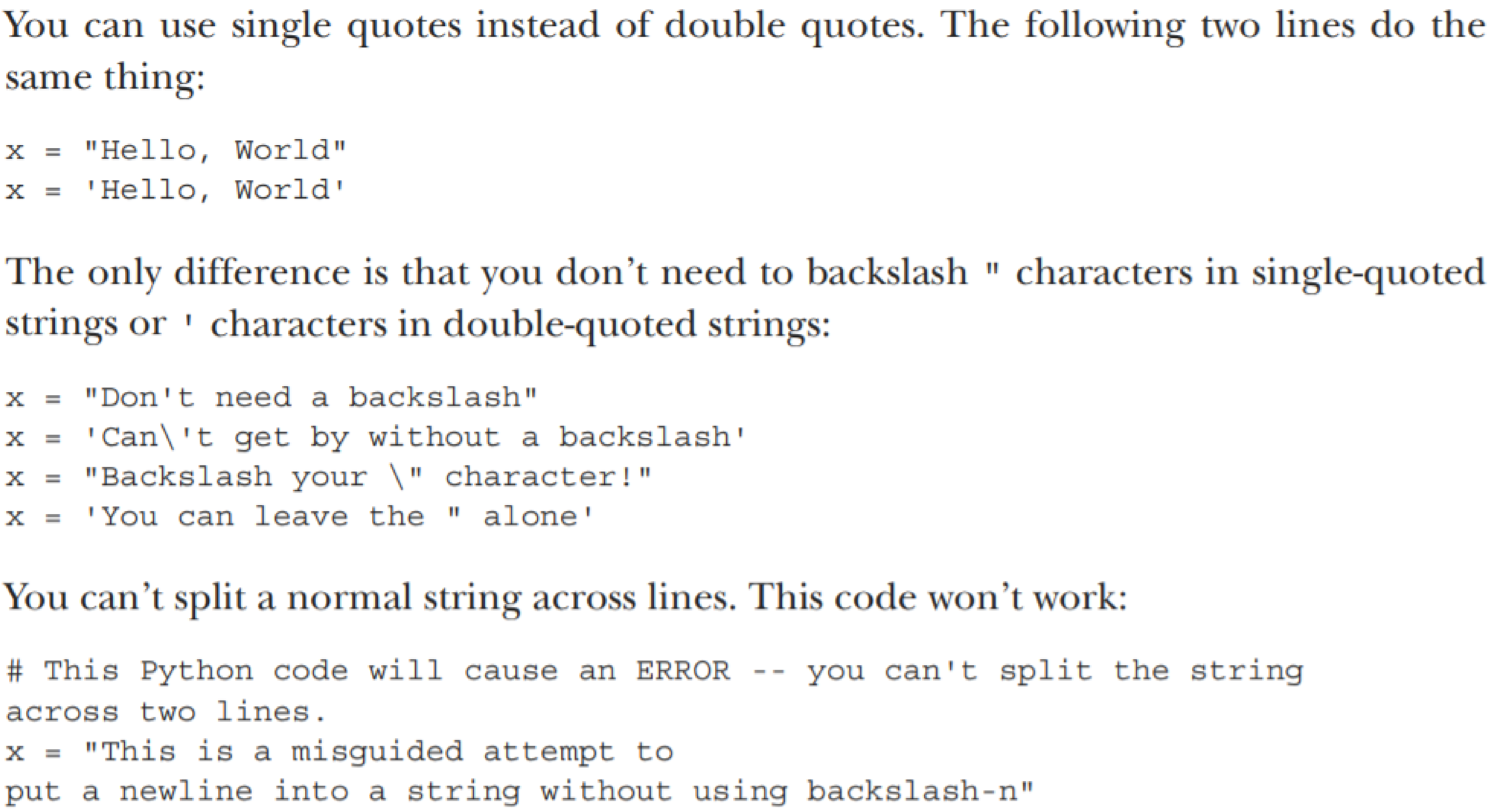
    \t means the tab character

    \\ means a single normal backslash character

    \" is a plain double-quote character

        It doesn’t end the string

```
x = "\tThis string starts with a \"tab\"."
x = "This string contains a single backslash(\\)."
```



You can use single quotes instead of double quotes. The following two lines do the same thing:

```
x = "Hello, World"
x = 'Hello, World'
```

The only difference is that you don’t need to backslash `"` characters in single-quoted strings or `'` characters in double-quoted strings:

```
x = "Don't need a backslash"
x = 'Can\'t get by without a backslash'
x = "Backslash your \" character!"
x = 'You can leave the " alone'
```

You can’t split a normal string across lines. This code won’t work:

```
# This Python code will cause an ERROR -- you can't split the string
across two lines.
x = "This is a misguided attempt to
put a newline into a string without using backslash-n"
```

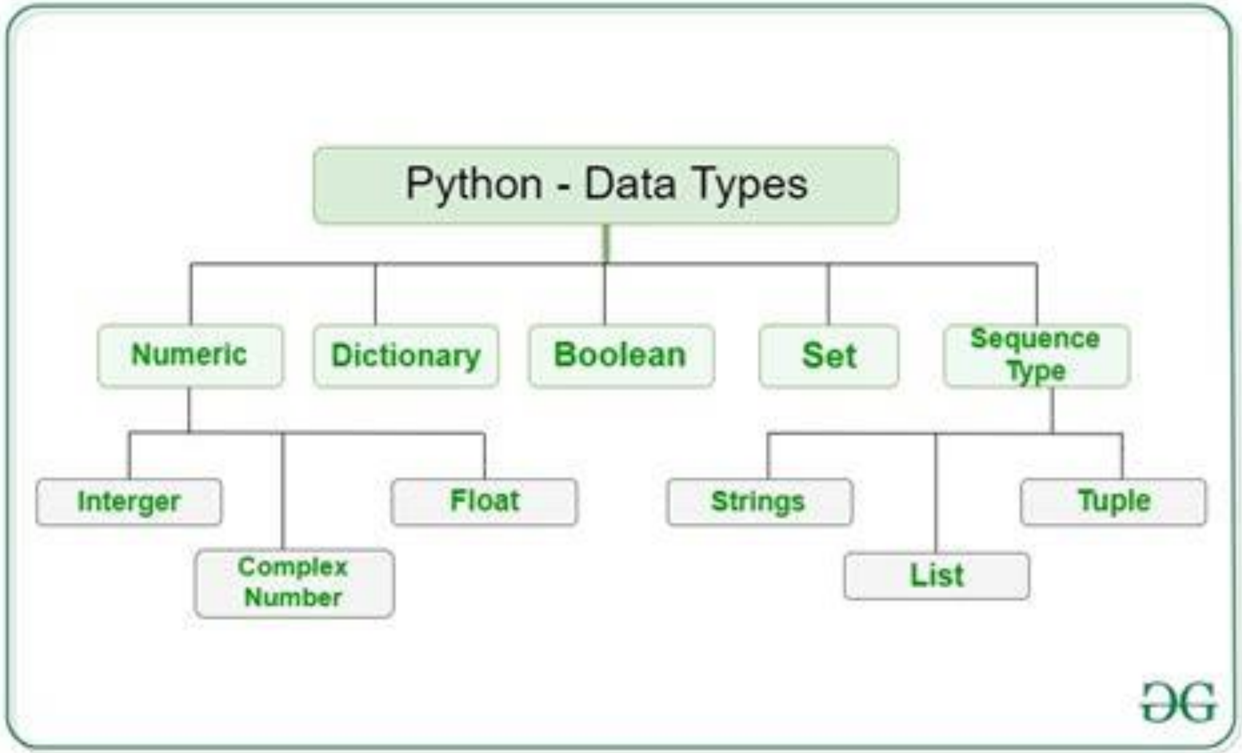
Strings

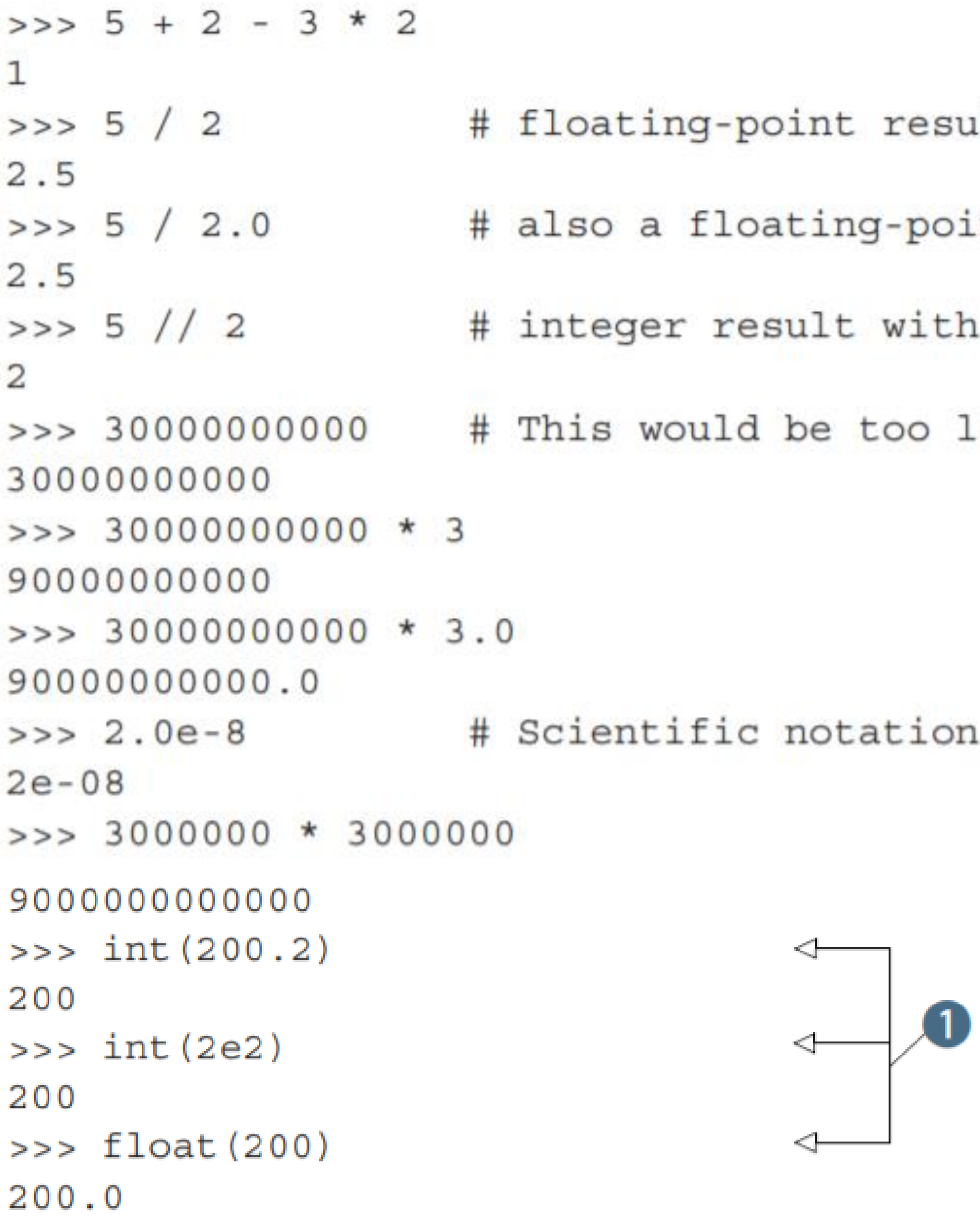
- But Python offers triple-quoted strings, which let you do this and include single and double quotes without backslashes:

```
x = """Starting and ending a string with triple " characters
permits embedded newlines, and the use of " and ' without
backslashes"""
```

Numbers

- Python offers four kinds of numbers: integers, floats, complex numbers, and Booleans





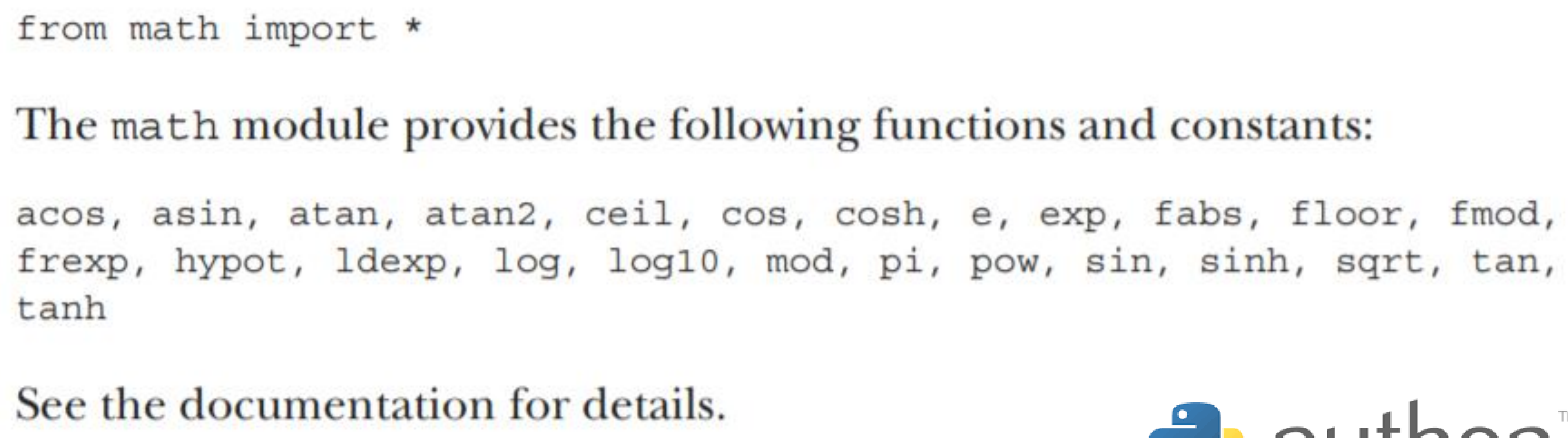
```
>>> 5 + 2 - 3 * 2
1
>>> 5 / 2           # floating-point result with normal division
2.5
>>> 5 / 2.0         # also a floating-point result
2.5
>>> 5 // 2          # integer result with truncation when divided using '//'
2
>>> 30000000000     # This would be too large to be an int in many languages
30000000000
>>> 30000000000 * 3
90000000000
>>> 30000000000 * 3.0
90000000000.0
>>> 2.0e-8          # Scientific notation gives back a float
2e-08
>>> 3000000 * 3000000
9000000000000
>>> int(200.2)
200
>>> int(2e2)
200
>>> float(200)
200.0
```

Built-in numeric functions

- Python provides the following number-related functions as part of its core:

abs, divmod, float, hex, int, max, min, oct, pow, round

Advanced numeric functions



```
from math import *
```

The `math` module provides the following functions and constants:

`acos, asin, atan, atan2, ceil, cos, cosh, e, exp, fabs, floor, fmod, frexp, hypot, ldexp, log, log10, mod, pi, pow, sin, sinh, sqrt, tan, tanh`

See the documentation for details.

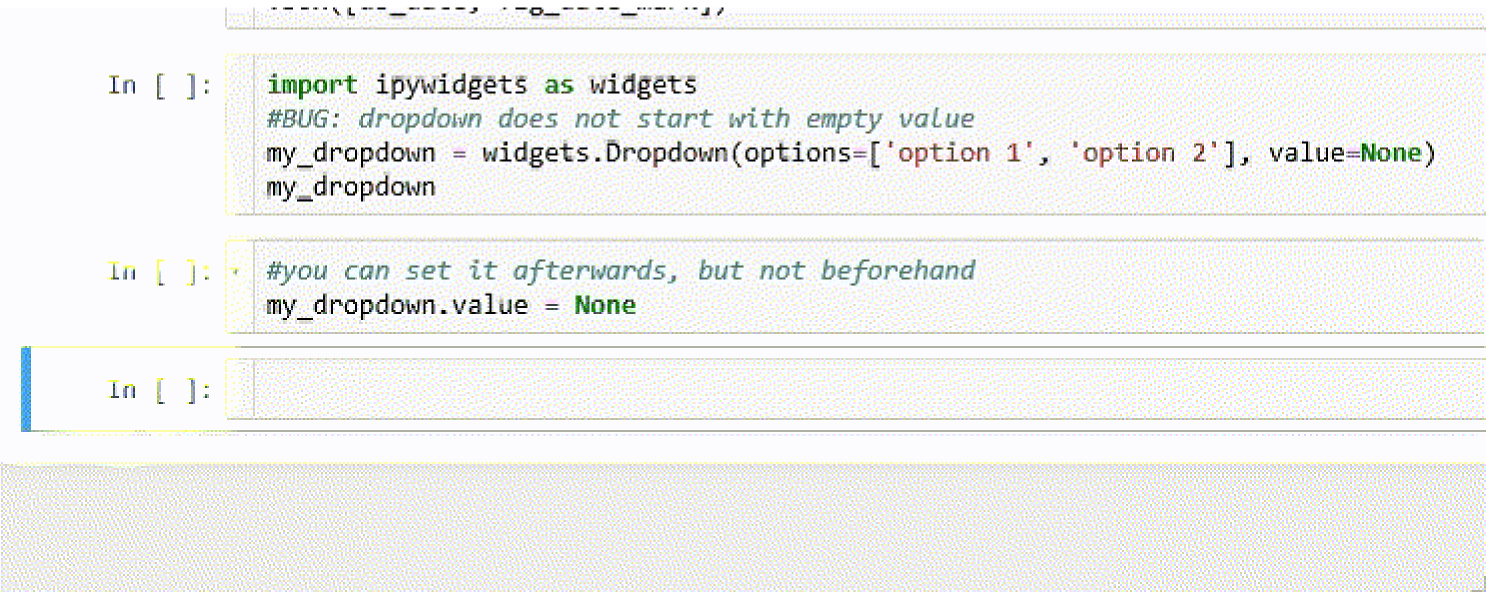
Numeric computation

- The core Python installation isn’t well suited to intensive numeric computation because of speed constraints.

- But the powerful Python extension NumPy provides highly efficient implementations of many advanced numeric operations
- (See: scipy.org and pandas – great for analytics!)

The None value

- None is used to represent an empty value.



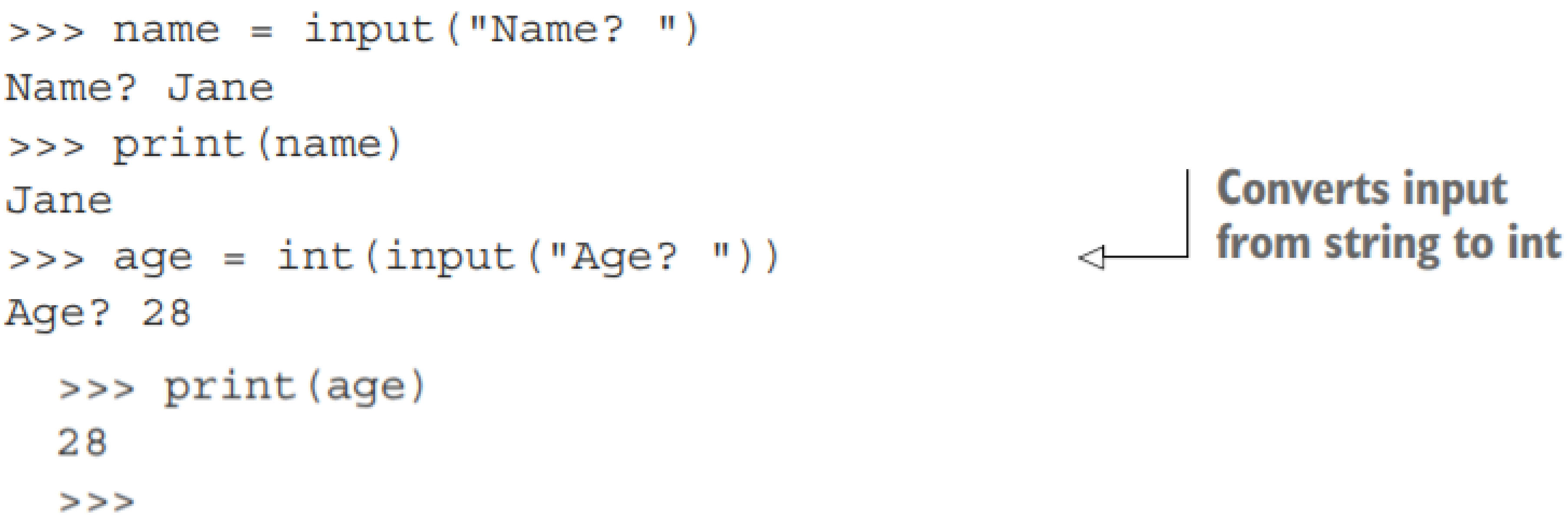
```
In [ ]: import ipywidgets as widgets
        #BUG: dropdown does not start with empty value
        my_dropdown = widgets.Dropdown(options=['option 1', 'option 2'], value=None)
        my_dropdown
```

```
In [ ]: #you can set it afterwards, but not beforehand
        my_dropdown.value = None
```

```
In [ ]:
```

Getting input from the user

- You can also use the input() function to get input from the user.
- Use the prompt string you want to display to the user as input's parameter:



```
>>> name = input("Name? ")
Name? Jane
>>> print(name)
Jane
>>> age = int(input("Age? "))
Age? 28
  >>> print(age)
  28
  >>>
```

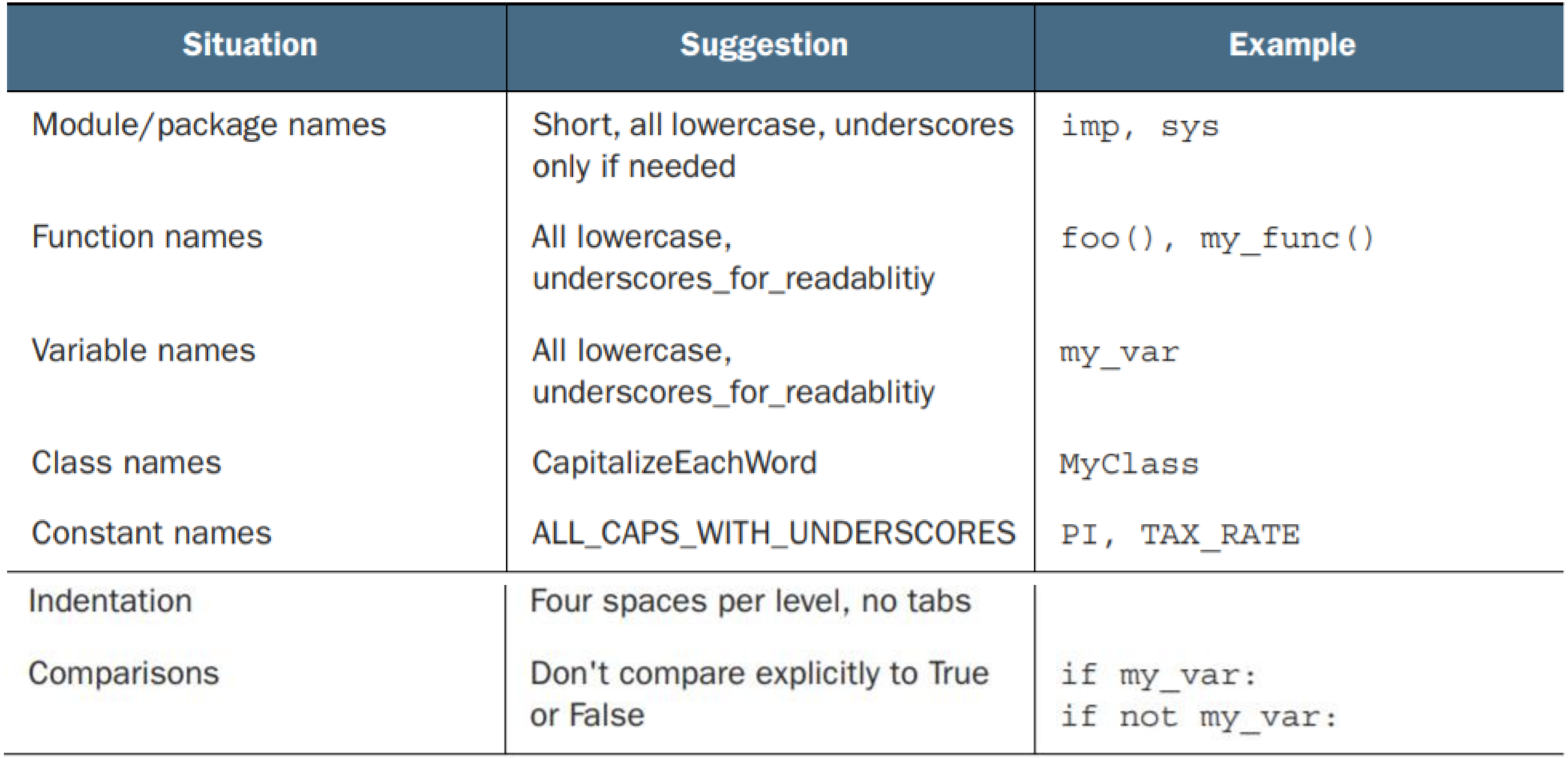
Converts input from string to int

Built-in operators

- Python provides various built-in operators, from the standard (+, *, and so on) to the more esoteric, such as operators for performing bit shifting, bitwise logical functions, and so forth.

- Most of these operators are no more unique to Python than to any other language; hence, I won’t explain them in the main text.

Basic Python style



Situation	Suggestion	Example
Module/package names	Short, all lowercase, underscores only if needed	`imp, sys`
Function names	All lowercase, underscores_for_readablitiy	`foo(), my_func()`
Variable names	All lowercase, underscores_for_readablitiy	`my_var`
Class names	CapitalizeEachWord	`MyClass`
Constant names	ALL_CAPS_WITH_UNDERSCORES	`PI, TAX_RATE`
Indentation	Four spaces per level, no tabs	

Comparisons	Don't compare explicitly to True or False	`if my_var:`
		`if not my_var:`

Basic Python style

- QUICK CHECK: PYTHONIC STYLE Which of the following variable and function names do you think are not good Pythonic style? Why?

bar(),

varName,

VERYLONGVARNAME,

foobar,

longvarname,

foo_bar(),

really_very_long_var_name

Quick Review

- The basic syntax summarized above is enough to start writing Python code.
- Python syntax is predictable and consistent.
- Because the syntax offers few surprises, many programmers can get started writing code surp quickly.