COMP-SCI 5590 - 0001 Special Topics

Introduction To Python

A readable, dynamic, pleasant, flexible, fast and powerful language

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

General Information

- ✓ Unlike C/C++ or Java, Python statements do not end in a semicolon
- ✓ In Python, indentation is the way you indicate the scope of a conditional, function, etc.
- ✓ Look, no braces!
- ✓ Python is interpretive.
- ✓ You can just enter statements into the Python environment and they'll execute

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Why do people use Python…?

✓ The following primary factors cited by Python users seem to be these:

✓ Python is object-oriented

✓ Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance.

✓ Indentation

✓ Indentation is one of the greatest feature in Python.

✓ It's free (open source)

✓ Downloading and installing Python is free and easy

✓ Source code is easily accessible

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

✓ It's powerful

- ✓ - Dynamic typing
- ✓ - Built-in types and tools
- ✓ - Library utilities
- ✓ - Third party utilities (e.g. Numeric, NumPy, SciPy)
- ✓ - Automatic memory management

✓ It's portable

- ✓ - Python runs virtually every major platform used today
- ✓ - As long as you have a compatible Python interpreter installed, Python programs will run in exactly the same manner, irrespective of platform.

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

- ✓ It's mixable
  - ✓ Python can be linked to components written in other languages easily
  - ✓ Linking to fast, compiled code is useful to computationally intensive problems
  - ✓ Python/C integration is quite common
- ✓ It's easy to use
  - ✓ No intermediate compile and link steps as in C/ C++
  - ✓ Python programs are compiled automatically to an intermediate form called bytecode, which the interpreter then reads
  - ✓ This gives Python the development speed of an interpreter without the performance loss inherent in purely interpreted languages
- ✓ It's easy to learn
  - ✓ Structure and syntax are pretty intuitive and easy to grasp

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Applications

- Science

  - Bioinformatics
- System Administration

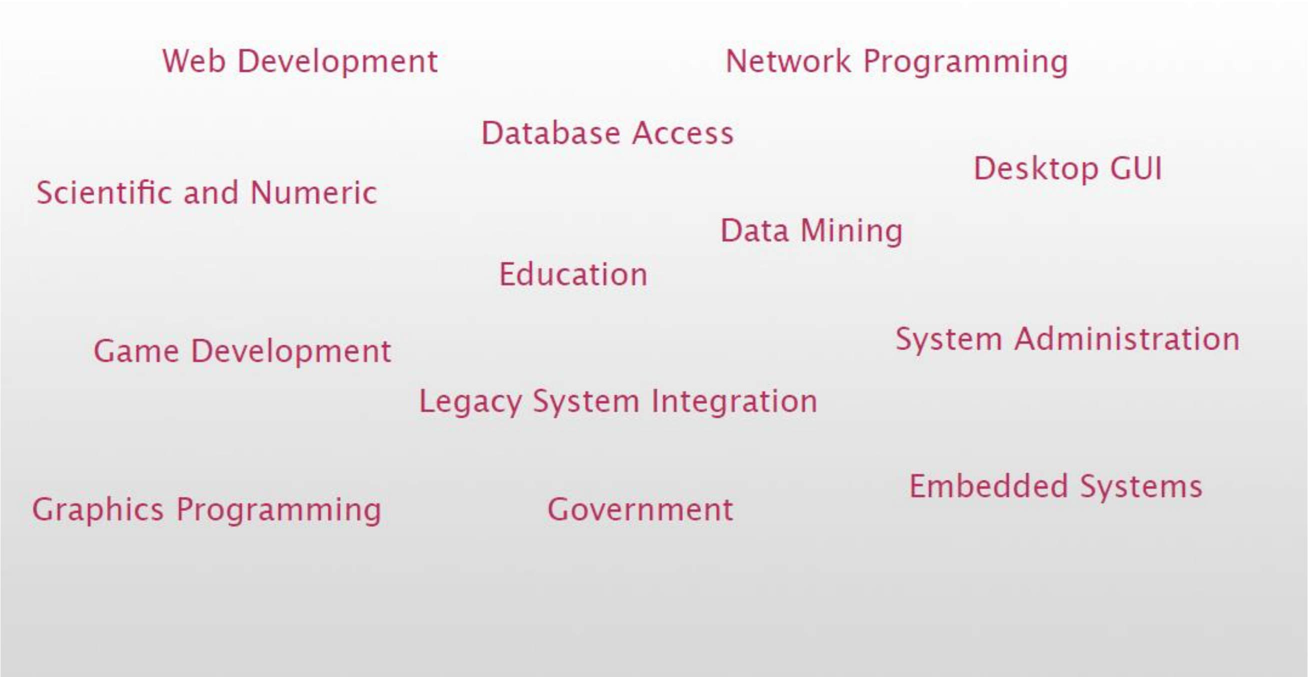
  -Unix

  -Web logic

  -Web sphere
- Web Application Development

  -CGI

  -Jython – Servlets
- Testing scripts



Web Development

Network Programming

Database Access

Desktop GUI

Scientific and Numeric

Data Mining

Education

System Administration

Game Development

Legacy System Integration

Embedded Systems

Graphics Programming

Government

Who uses python today…

- Python is being applied in real revenue-generating products by real companies. For instance:
- Google makes extensive use of Python in its web search system, and employs Python's creator.
- Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm, and IBM use Python for hardware testing.
- ESRI uses Python as an end-user customization tool for its popular GIS mapping products.
- The YouTube video sharing service is largely written in Python
- The list goes on…….

slideshare
YouTube
Broadcast Yourself™
SurveyMonkey
reddit
DISQUS
Google
Dropbox
facebook
Instagram
mozilla
FOUNDATION
Pinterest
Quora
Spotify
NASA

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Who created Python?

- "My original motivation for creating Python was the perceived need for a higher level language in the Amoeba [Operating Systems] project.
- I realized that the development of system administration utilities in C was taking too long. Moreover, doing these things in the Bourne shell wouldn't work for a variety of reasons. ...

  So, there was a need for a language that would bridge the gap between C and the shell"

Guido Van Rossum

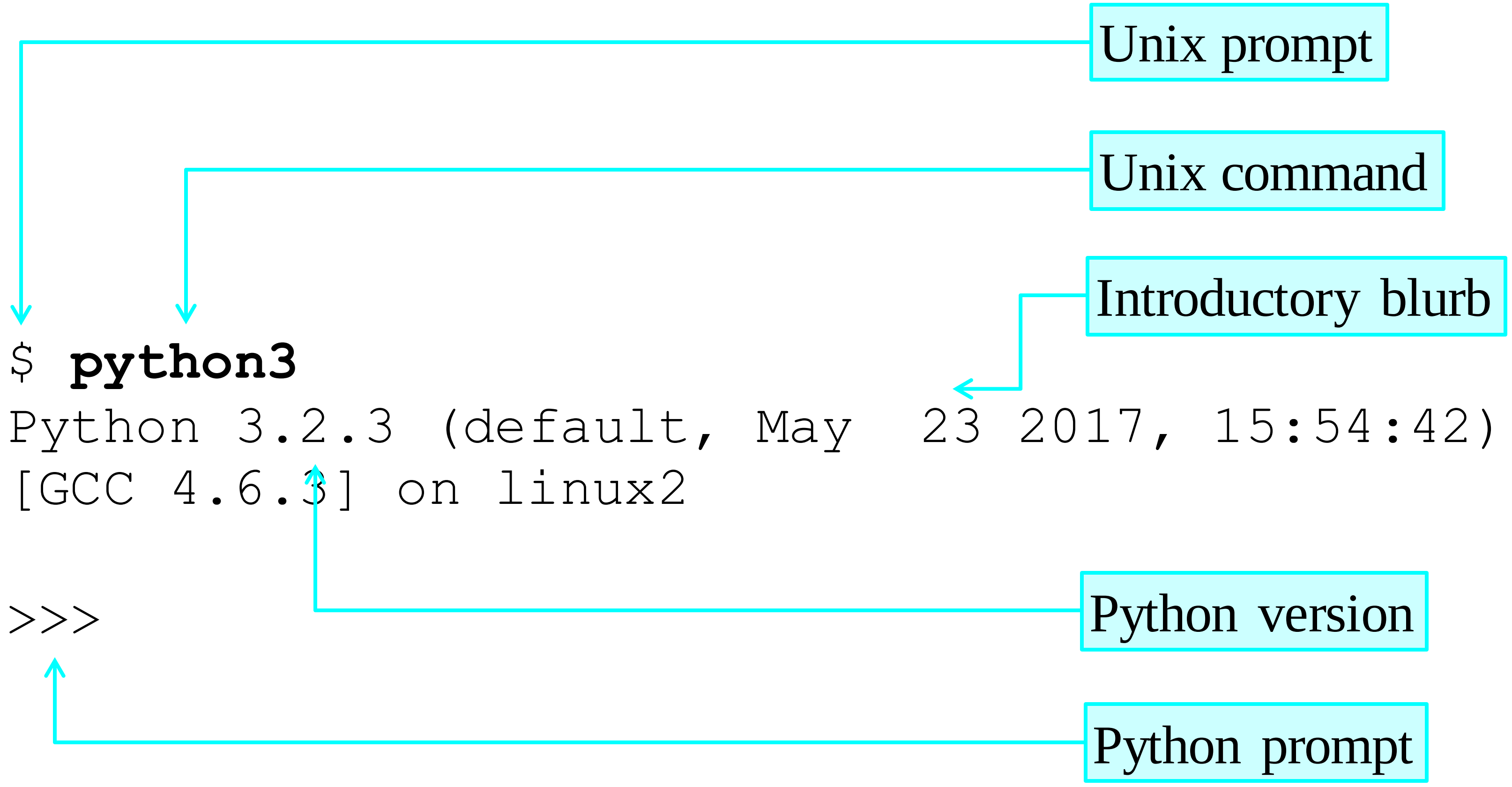
-The Creator

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Installation ??

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Running Python — 2 (UNIX)



```
$ python3
Python 3.2.3 (default, May  23 2017, 15:54:42)
[GCC 4.6.3] on linux2

>>>
```

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

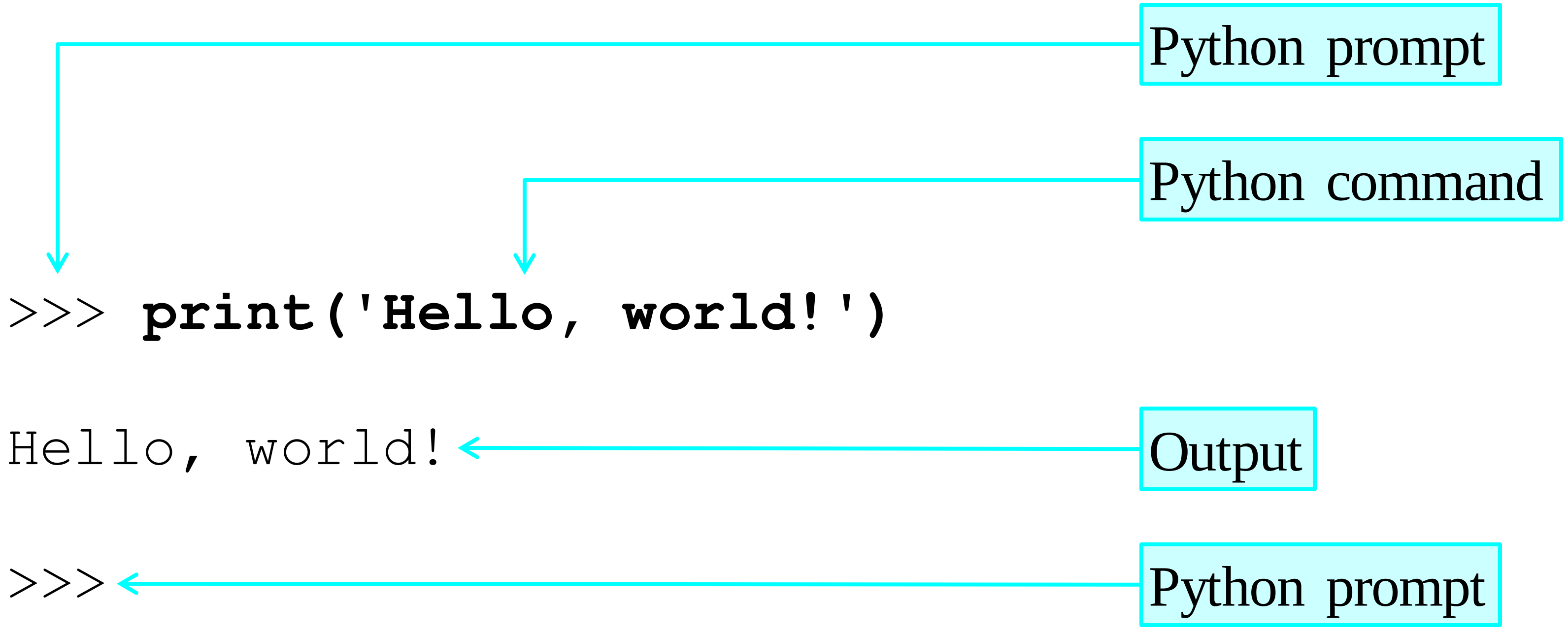
Quitting Python

```
>>> exit()

>>> quit()

>>> Ctrl + D
```

Any one of these

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

A first Python command



```
>>> print('Hello, world!')
Hello, world!
>>>
```

Python prompt

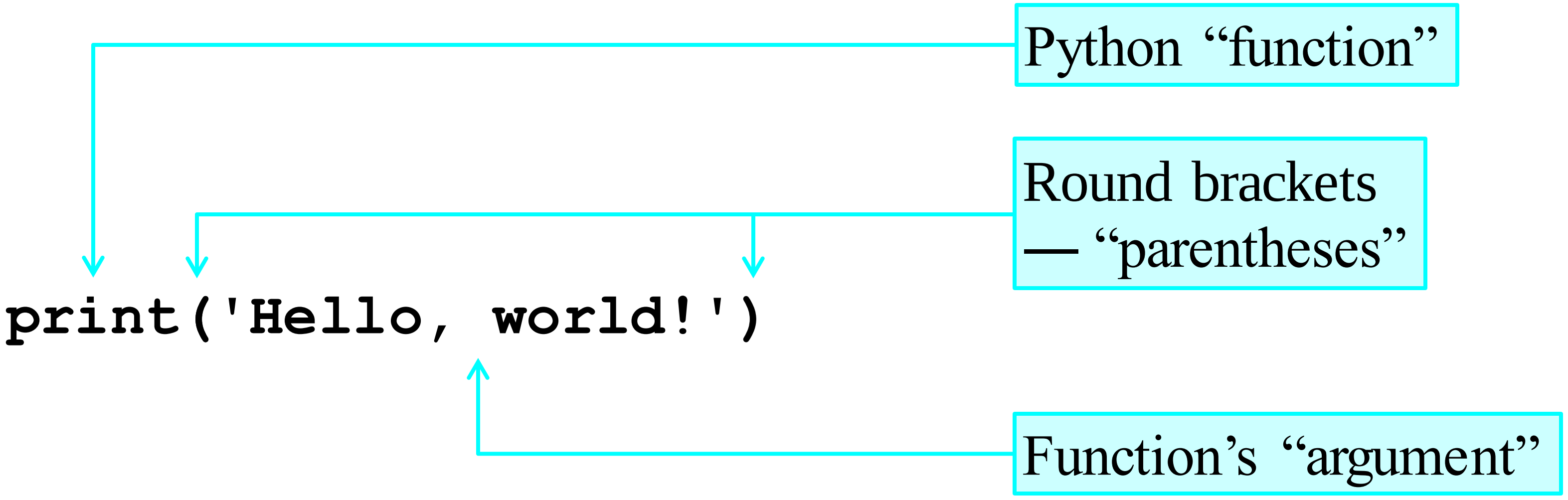
Python command

Output

Python prompt

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

Python commands



```
print('Hello, world!')
```

Python “function”

Round brackets — “parentheses”

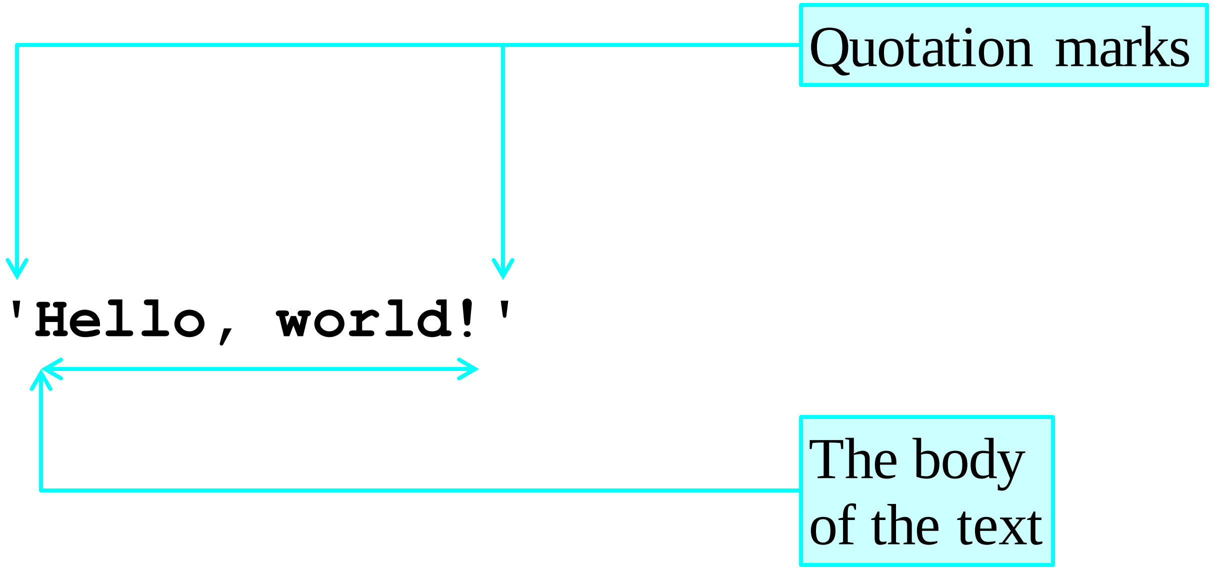
Function’s “argument”

```
print  ≠  PRINT
```

“Case sensitive”

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Python text



```
'Hello, world!'
```

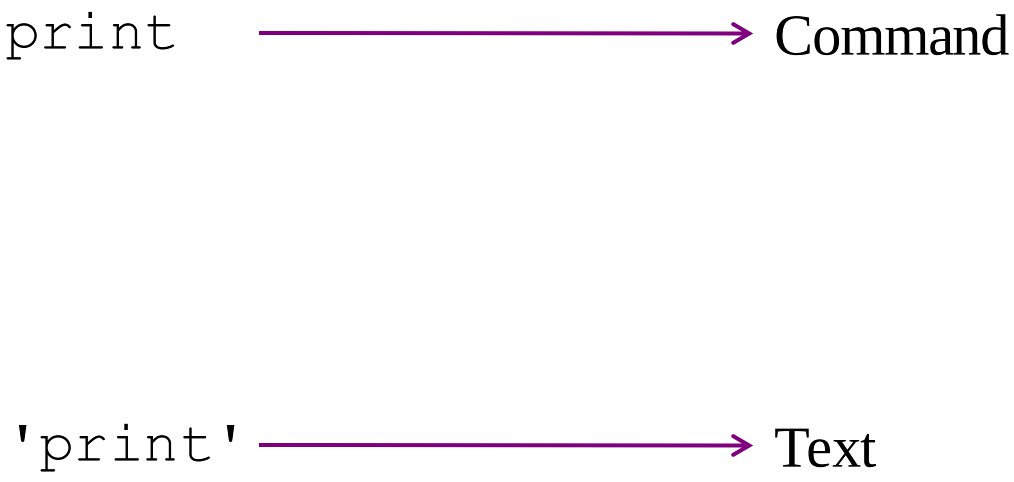
Quotation marks

The body of the text

The quotes are not part of the text itself.

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Quotes?



`print` → Command

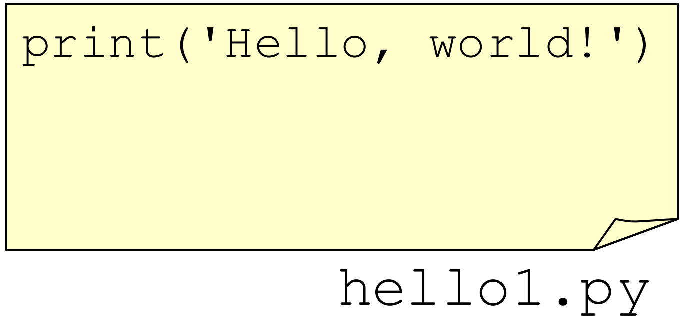
`'print'` → Text

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

Python scripts

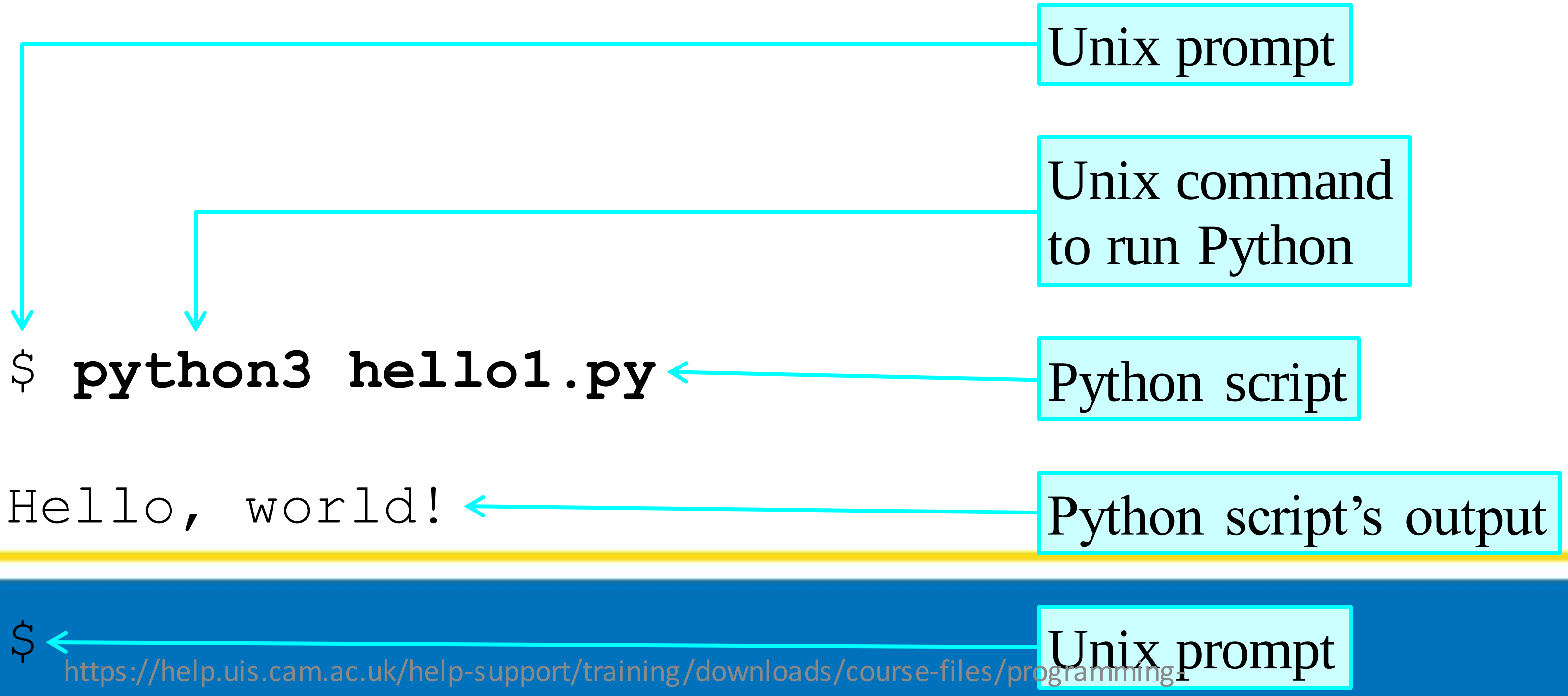
File in home directory

Run from *Unix* prompt



```
print('Hello, world!')
```

hello1.py

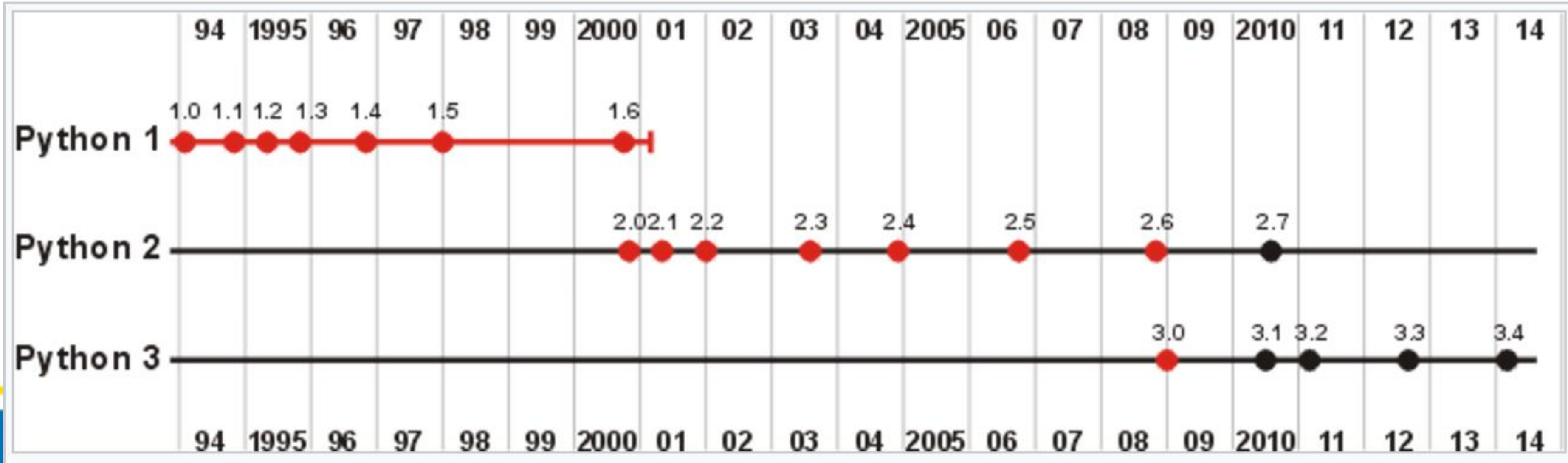


```
$ python3 hello1.py
Hello, world!
$
```

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

4 Major Versions of Python

- “Python” or “CPython” is written in C/C++
  - Version 2.7 came out in mid-2010
  - Version 3.4 came out in early 2014
- “Jython” is written in Java for the JVM
- “IronPython” is written in C# for the .Net environment

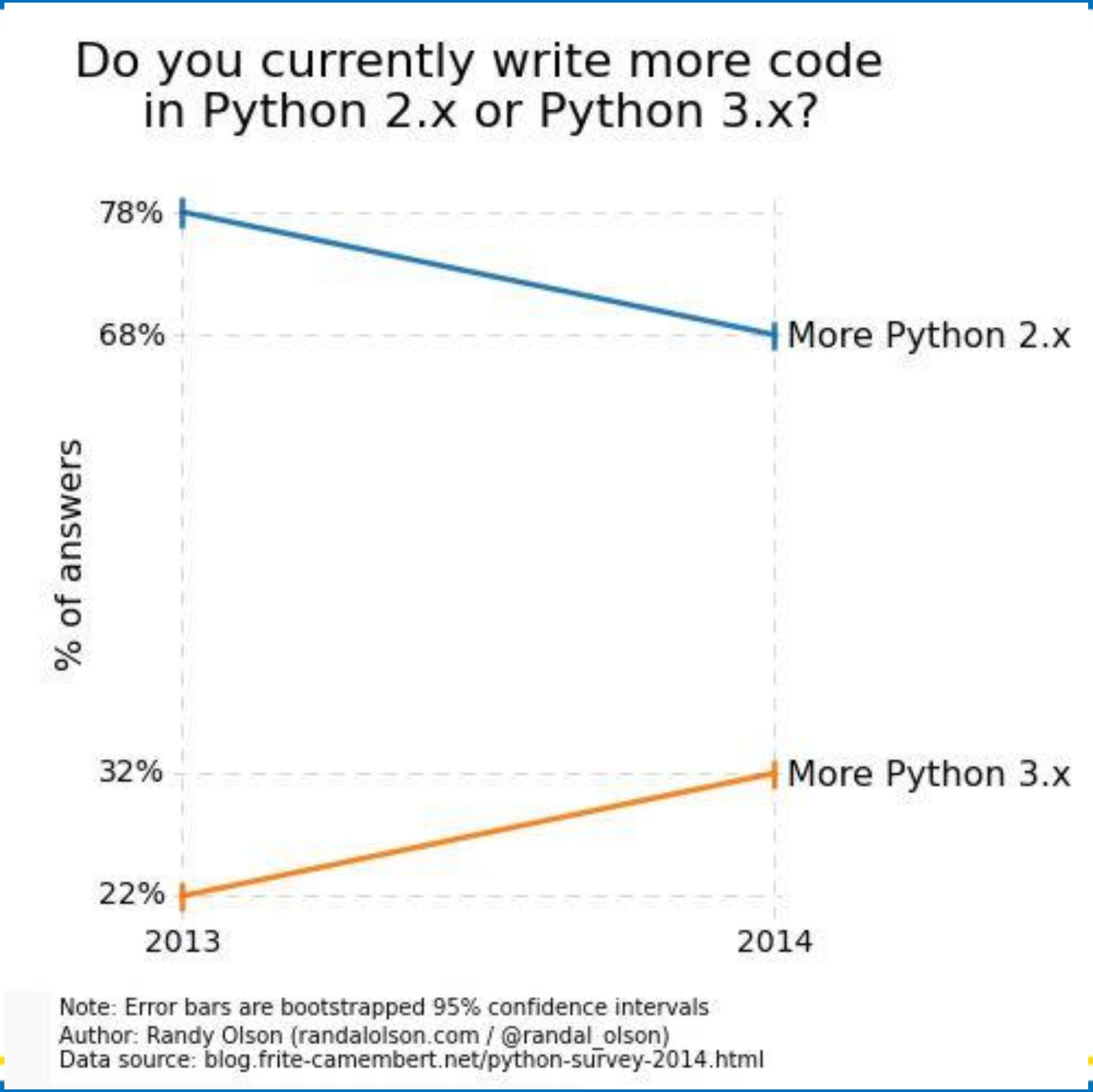


https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-26647807

Python 2 vs 3

- Learn more from below link:
- http://learntocodewith.me/programming/python/python-2-vs-python-3/



Note: Error bars are bootstrapped 95% confidence intervals
Author: Randy Olson (randalolson.com / @randal_olson)
Data source: blog.frite-camembert.net/python-survey-2014.html

Development Environments

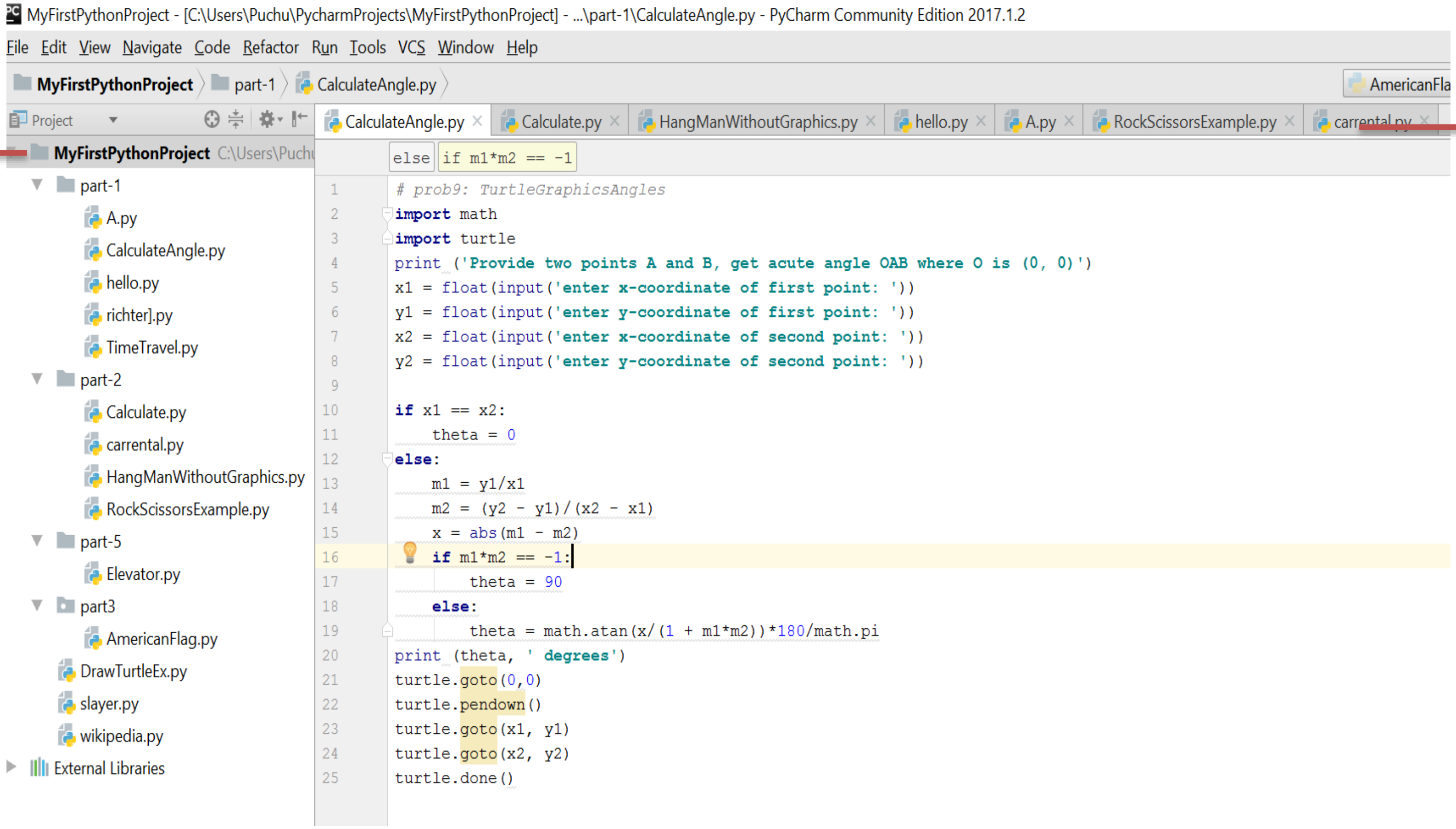
what IDE to use? http://stackoverflow.com/questions/81584

1. PyDev with Eclipse
2. Komodo
3. Emacs
4. Vim
5. TextMate
6. Gedit
7. Idle
8. PIDA (Linux)(VIM Based)
9. NotePad++ (Windows)
10. BlueFish (Linux)
11. PyCharm

Our choice of IDE !!

https://help.uis.cam.ac.uk/help-support/training/downloads/course-files/programming-student-files/python-courses/pythonab/pythonab-files/python3-slides.pdf

Pycharm IDE



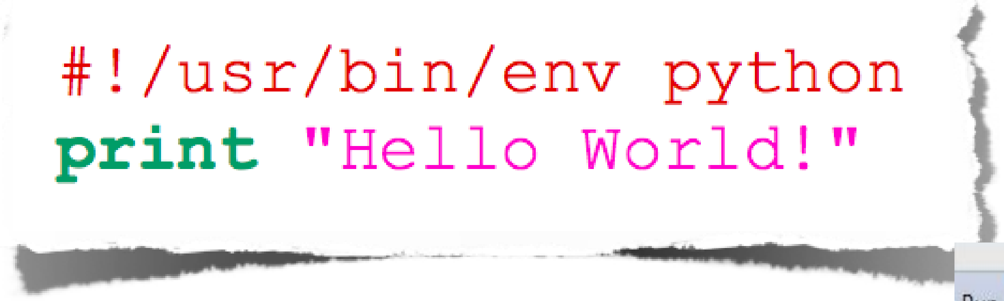
```python
# prob9: TurtleGraphicsAngles
import math
import turtle
print ('Provide two points A and B, get acute angle OAB where O is (0, 0)')
x1 = float(input('enter x-coordinate of first point: '))
y1 = float(input('enter y-coordinate of first point: '))
x2 = float(input('enter x-coordinate of second point: '))
y2 = float(input('enter y-coordinate of second point: '))

if x1 == x2:
    theta = 0
else:
    m1 = y1/x1
    m2 = (y2 - y1)/(x2 - x1)
    x = abs(m1 - m2)
    if m1*m2 == -1:
        theta = 90
    else:
        theta = math.atan(x/(1 + m1*m2))*180/math.pi
print (theta, ' degrees')
turtle.goto(0,0)
turtle.pendown()
turtle.goto(x1, y1)
turtle.goto(x2, y2)
turtle.done()
```

Project Name

Python File Name

Start >>>

Usecase 1 - Hello World

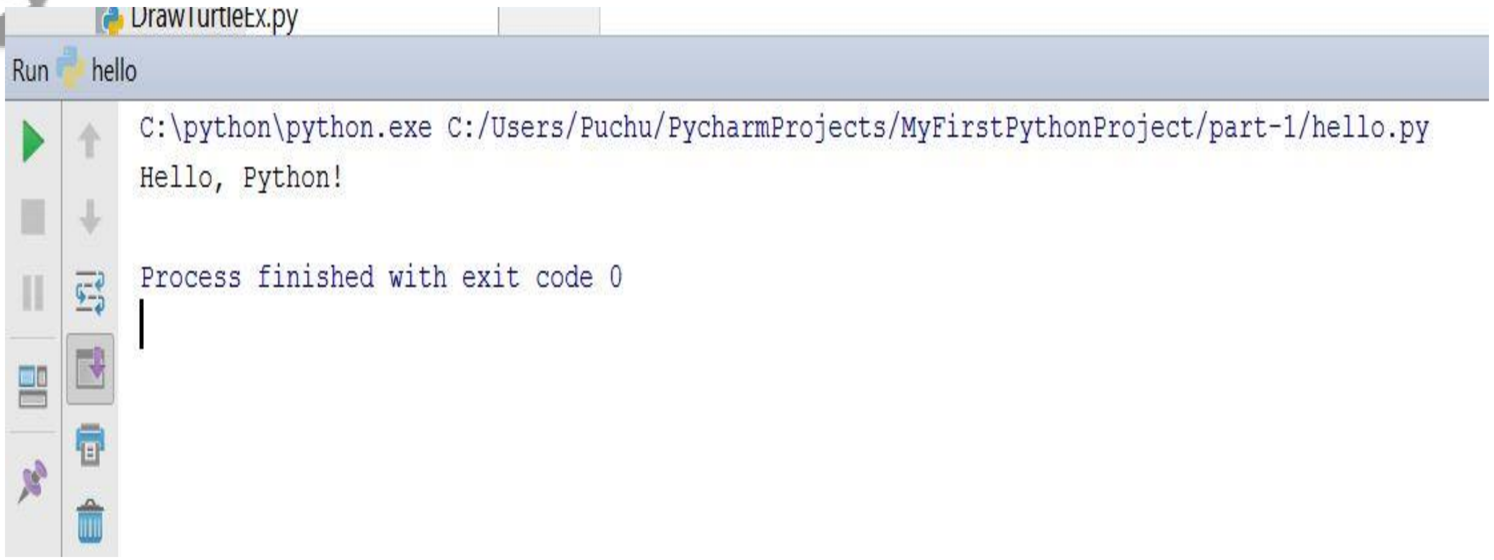


```
#!/usr/bin/env python
print "Hello World!"
```

hello_world.py

Input Sample Program

Output



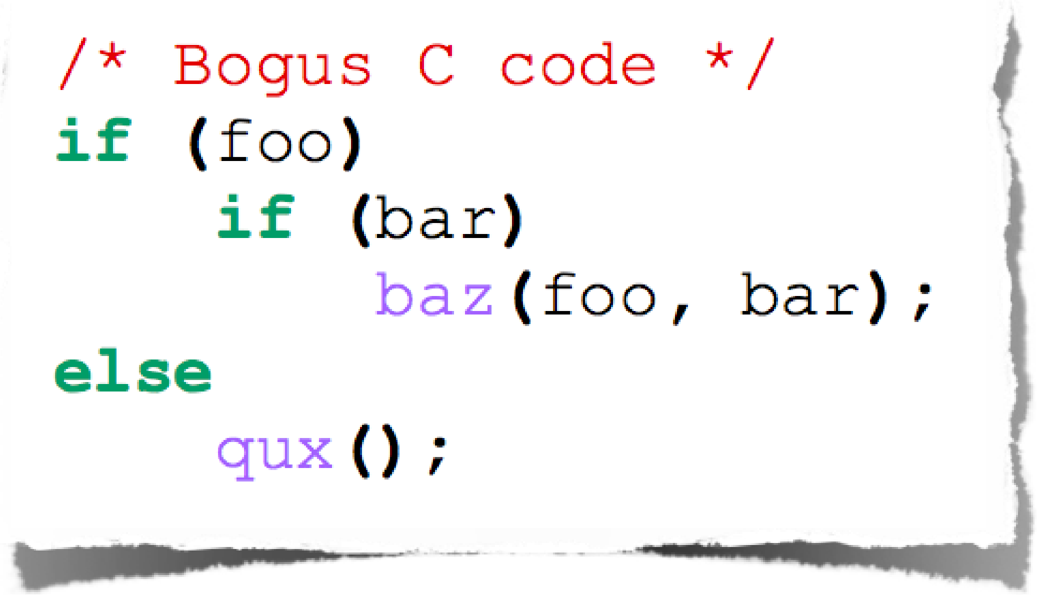
```
C:\python\python.exe C:/Users/Puchu/PycharmProjects/MyFirstPythonProject/part-1/hello.py
Hello, Python!

Process finished with exit code 0
```

Indentation

Most languages don’t care about indentation

Most humans do

We tend to group similar things together

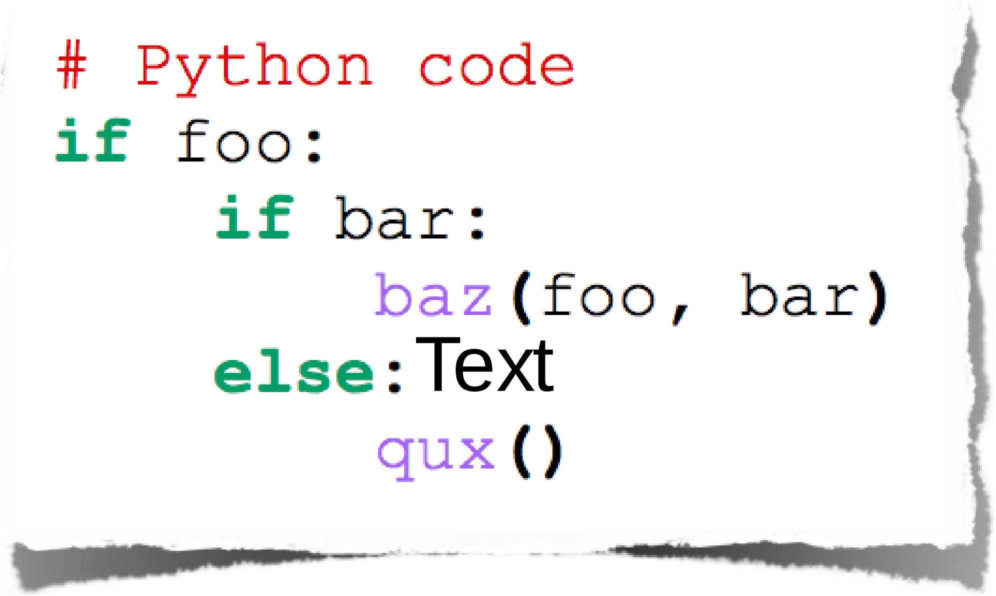


```
/* Bogus C code */
if (foo)
    if (bar)
        baz(foo, bar);
else
    qux();
```

The else here actually belongs to the 2nd if statement

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Indentation



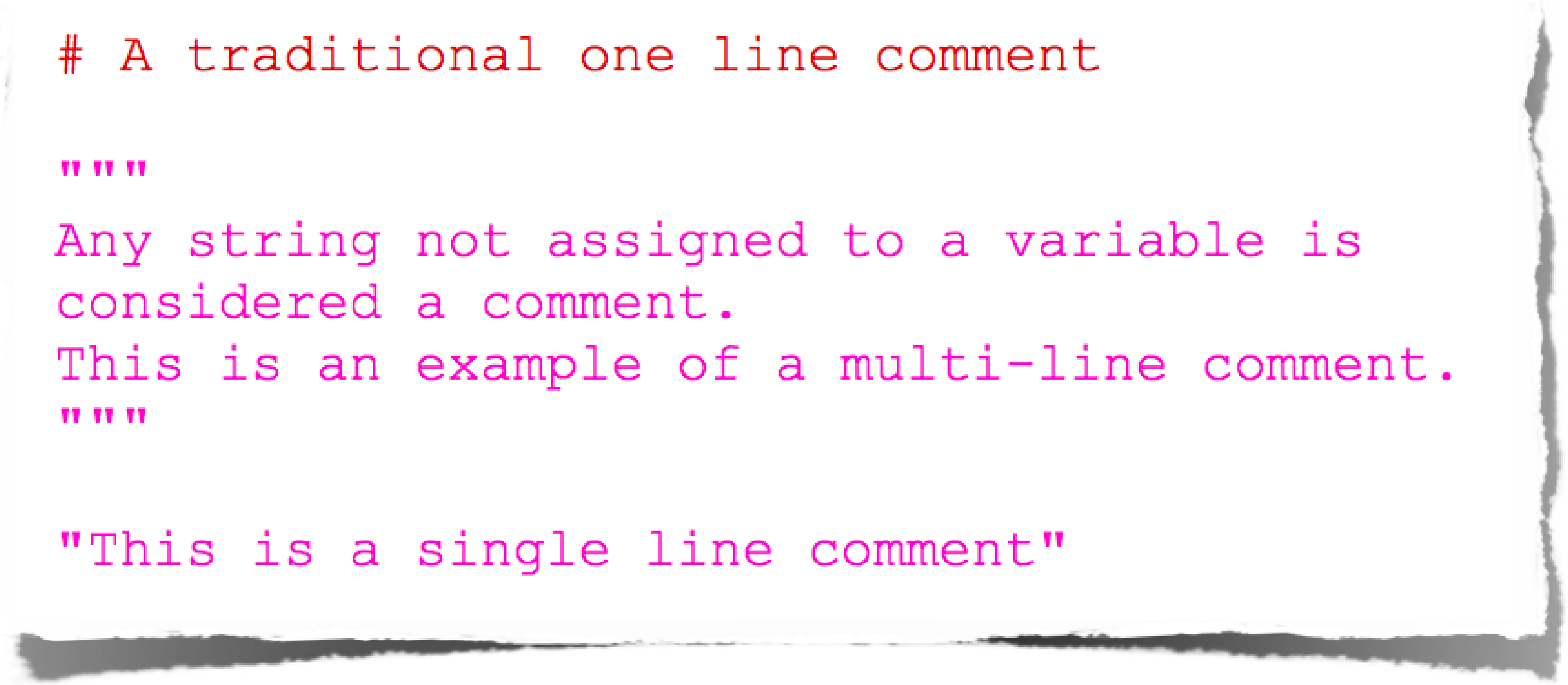
```
# Python code
if foo:
    if bar:
        baz(foo, bar)
    else:
        qux()
```

Text

Python embraces indentation

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Comments



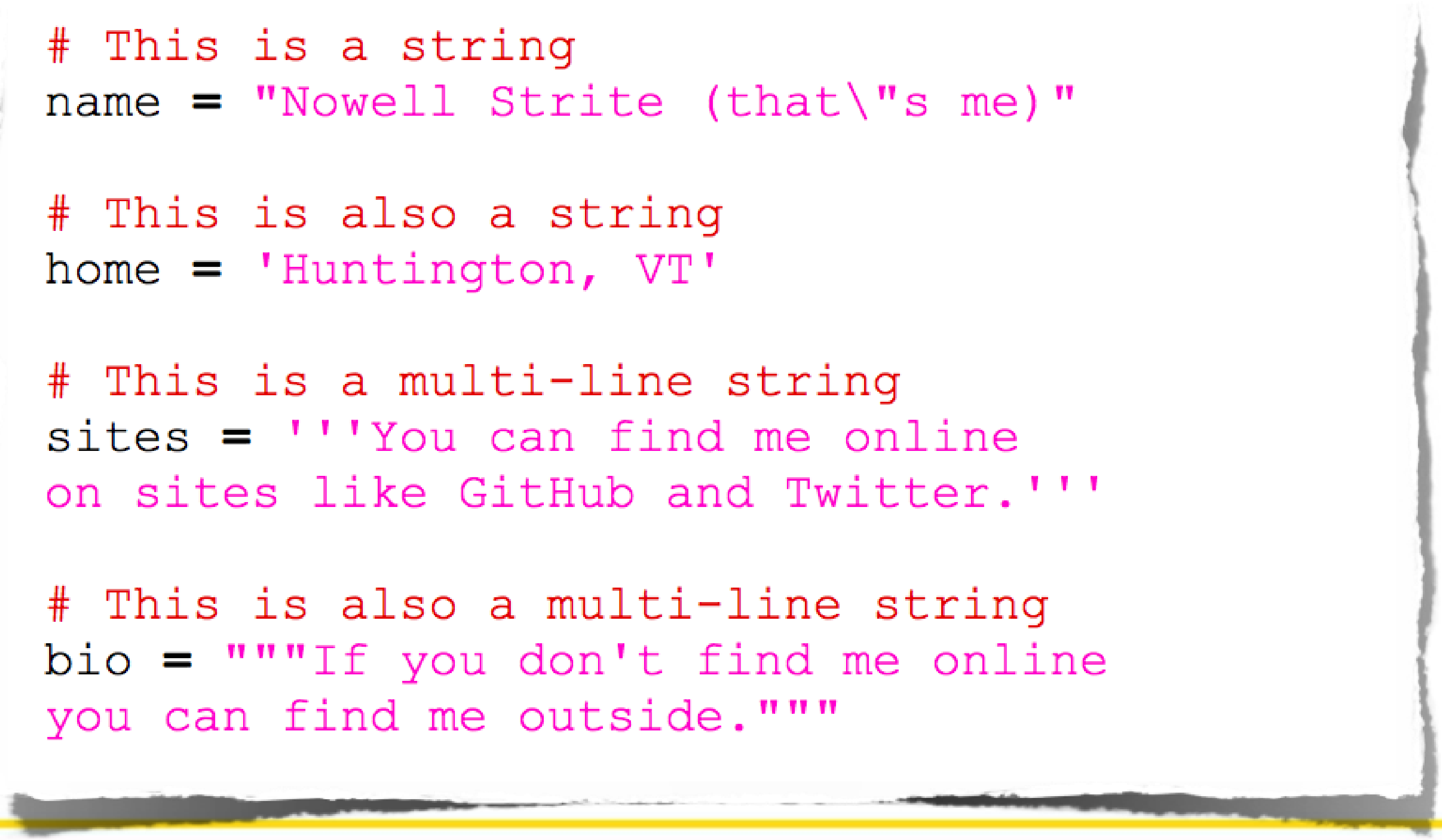
```
# A traditional one line comment

"""
Any string not assigned to a variable is
considered a comment.
This is an example of a multi-line comment.
"""

"This is a single line comment"
```

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

Types

https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807

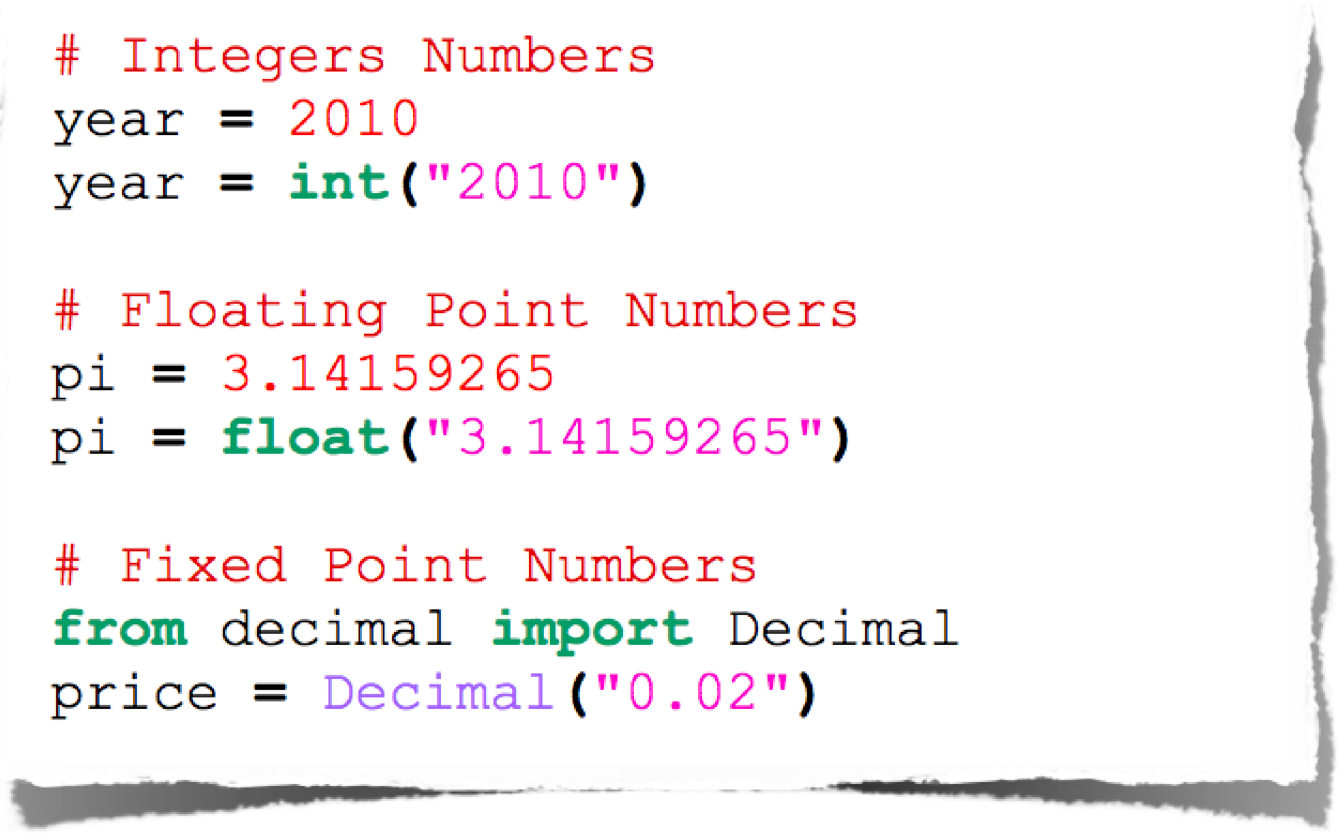
Strings



```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

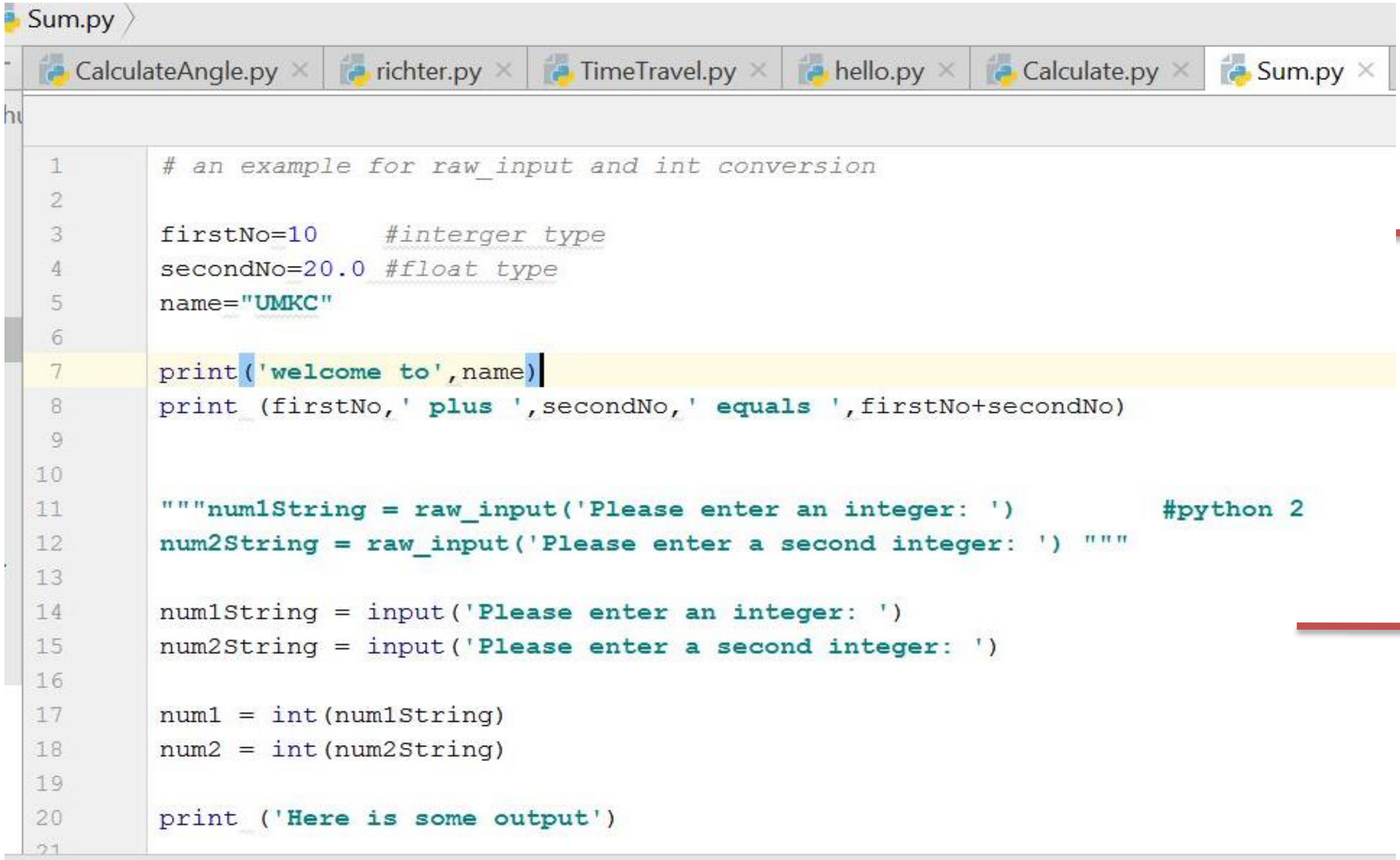
Numbers



```
# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")

# Fixed Point Numbers
from decimal import Decimal
price = Decimal("0.02")
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

Use case 2- Add two numbers



```
# an example for raw_input and int conversion

firstNo=10     #interger type
secondNo=20.0 #float type
name="UMKC"

print('welcome to',name)
print (firstNo,' plus ',secondNo,' equals ',firstNo+secondNo)


"""num1String = raw_input('Please enter an integer: ')          #python 2
num2String = raw_input('Please enter a second integer: ') """

num1String = input('Please enter an integer: ')
num2String = input('Please enter a second integer: ')

num1 = int(num1String)
num2 = int(num2String)

print ('Here is some output')
```

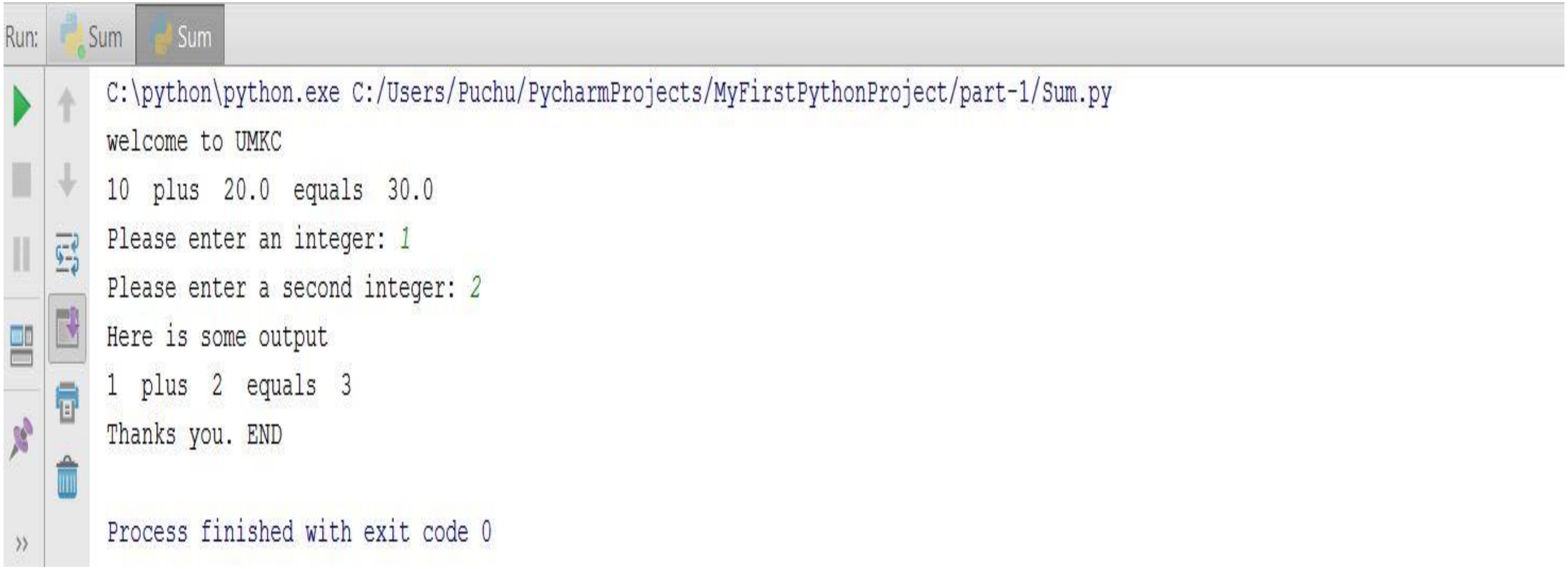
Variable declaration

Print statement

Python 2 way

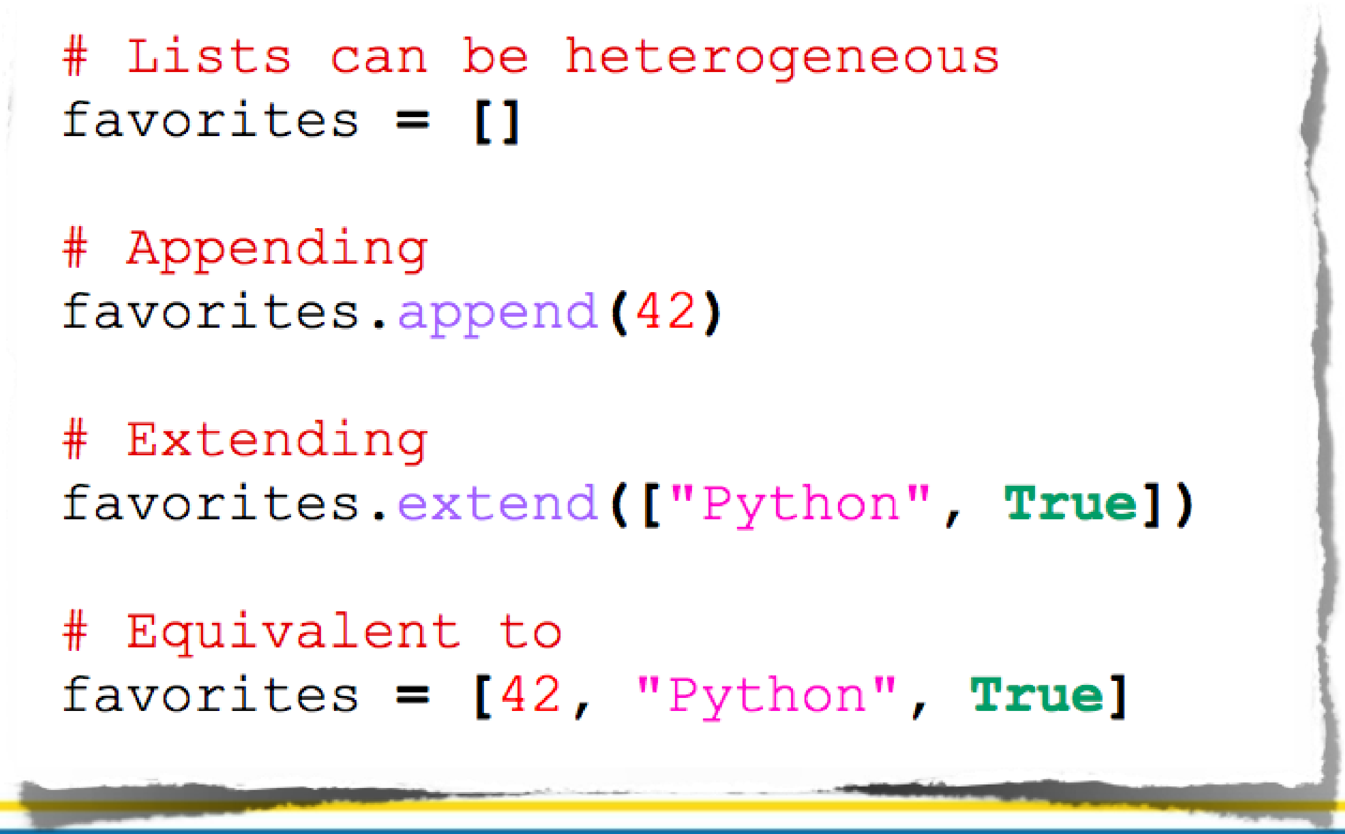
Taking Input from user

Use case 2- Output



```
C:\python\python.exe C:/Users/Puchu/PycharmProjects/MyFirstPythonProject/part-1/Sum.py
welcome to UMKC
10  plus  20.0  equals  30.0
Please enter an integer: 1
Please enter a second integer: 2
Here is some output
1  plus  2  equals  3
Thanks you. END

Process finished with exit code 0
```

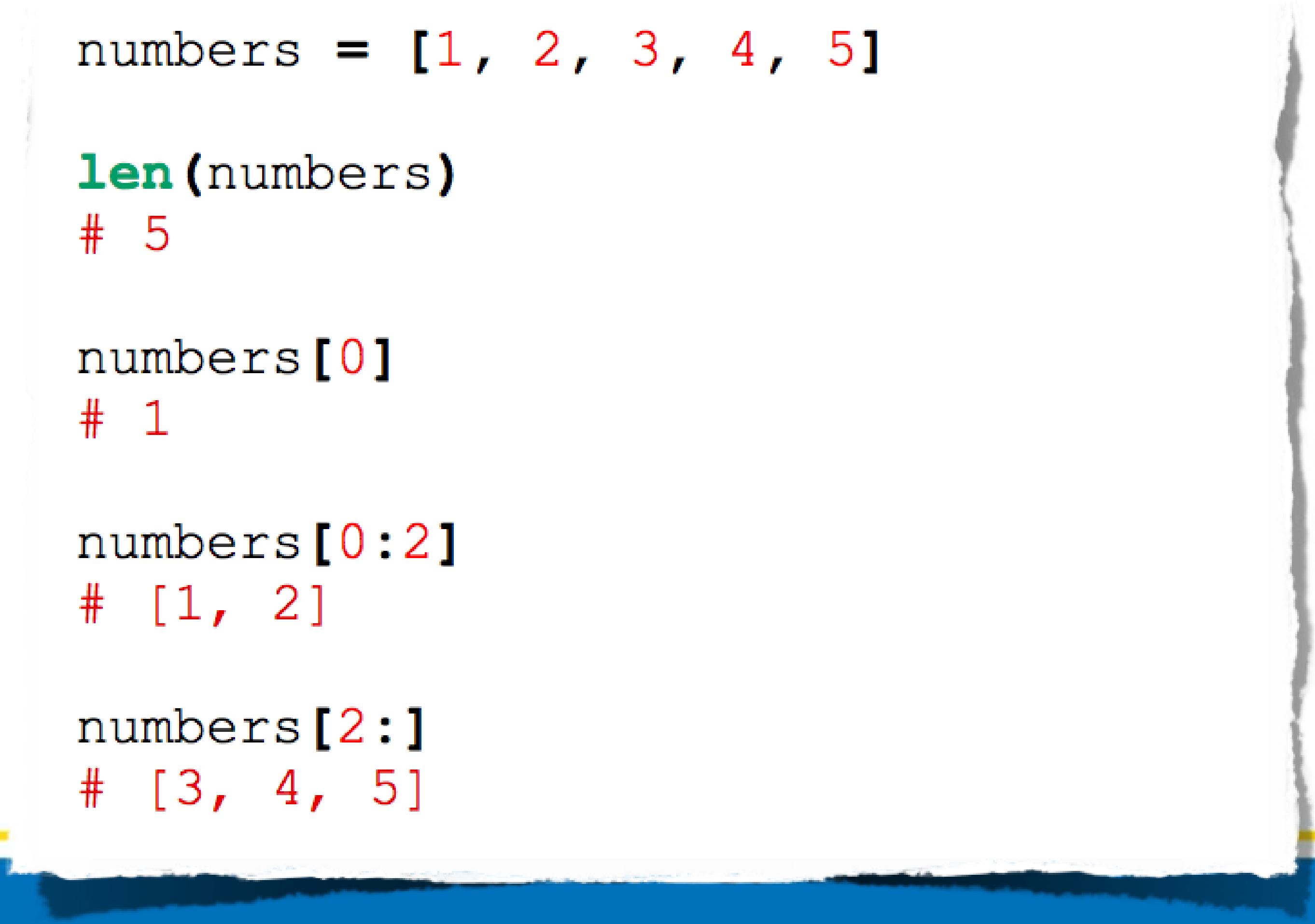
Lists



```
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

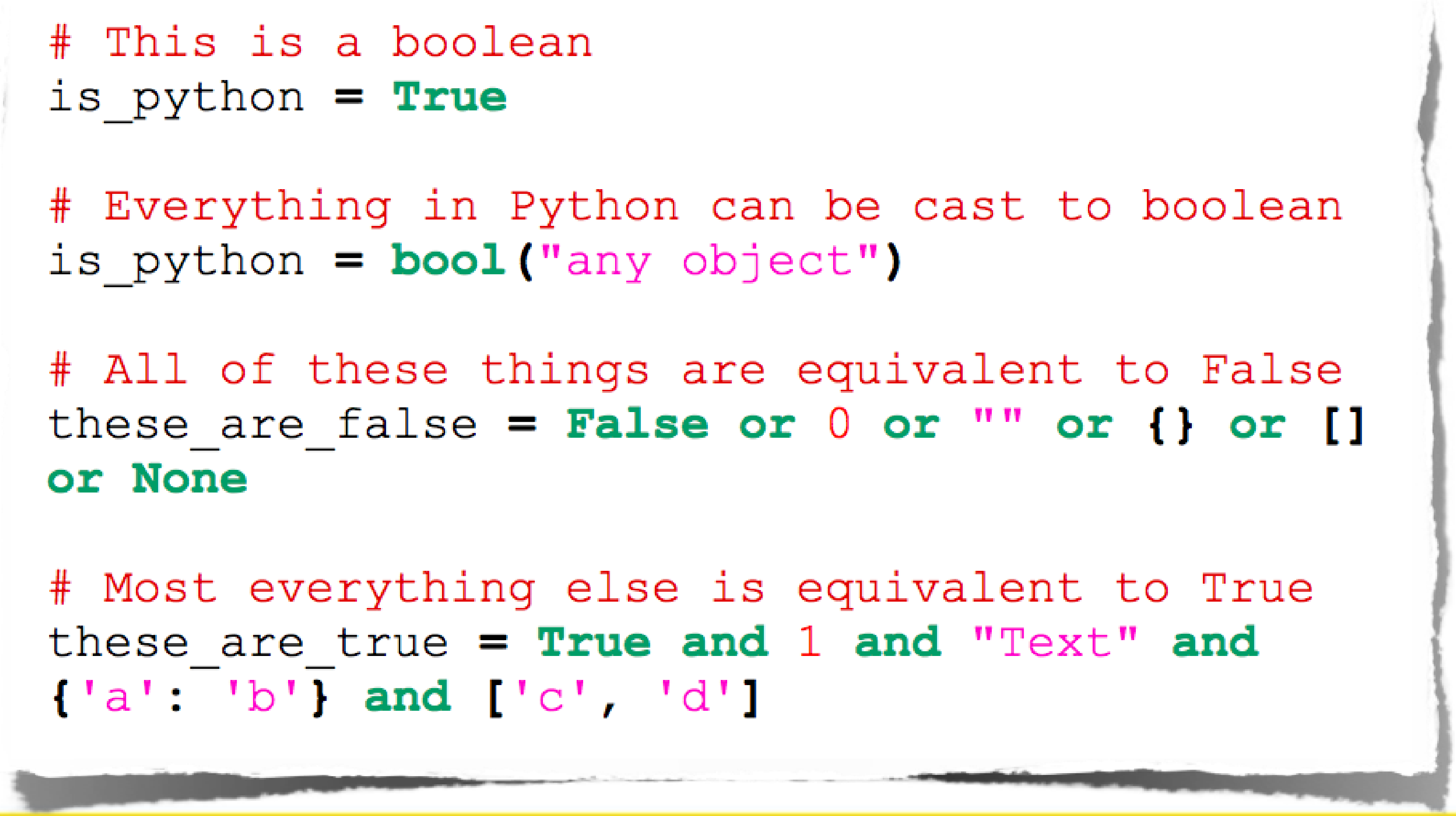
Lists



```
numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

Booleans



```
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```

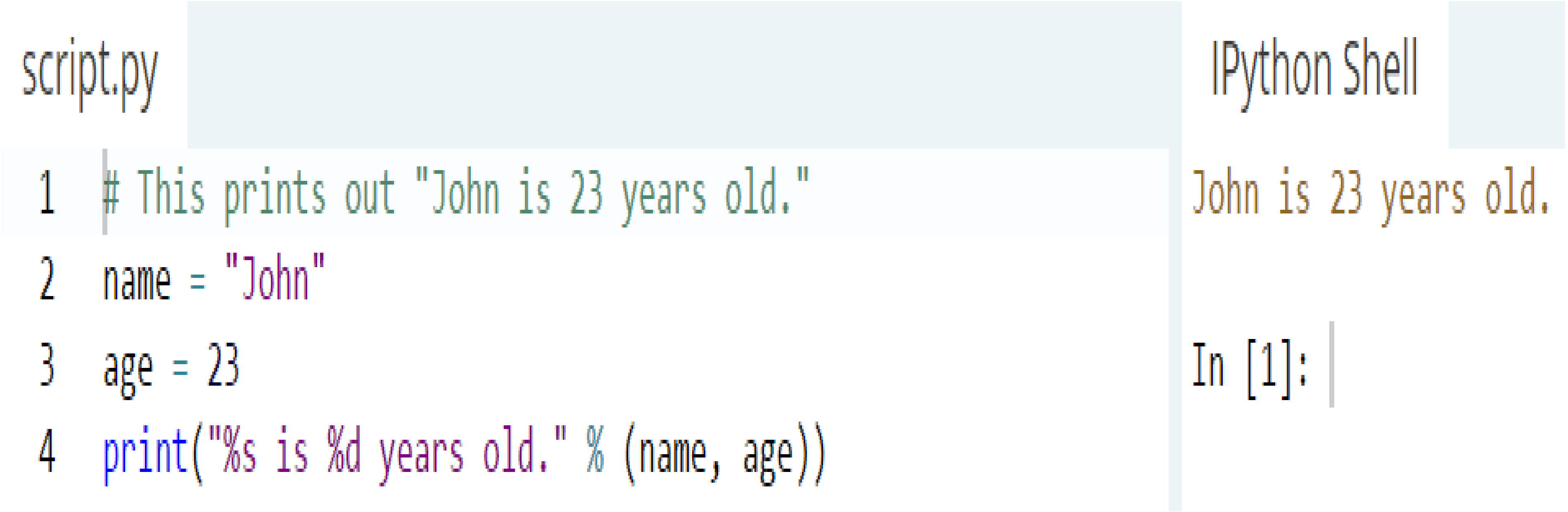
https://www.slideshare.net/nowells/introduction-to-python-5182313

Operators

String Formatting

- uses C-style string formatting to create new, formatted strings
- The "%" operator is used to format
- Let's say you have a variable called "name" with your user name in it, and you would then like to print

String Formatting: Example



```python
# This prints out "John is 23 years old."
name = "John"
age = 23
print("%s is %d years old." % (name, age))
```

```
John is 23 years old.

In [1]:
```

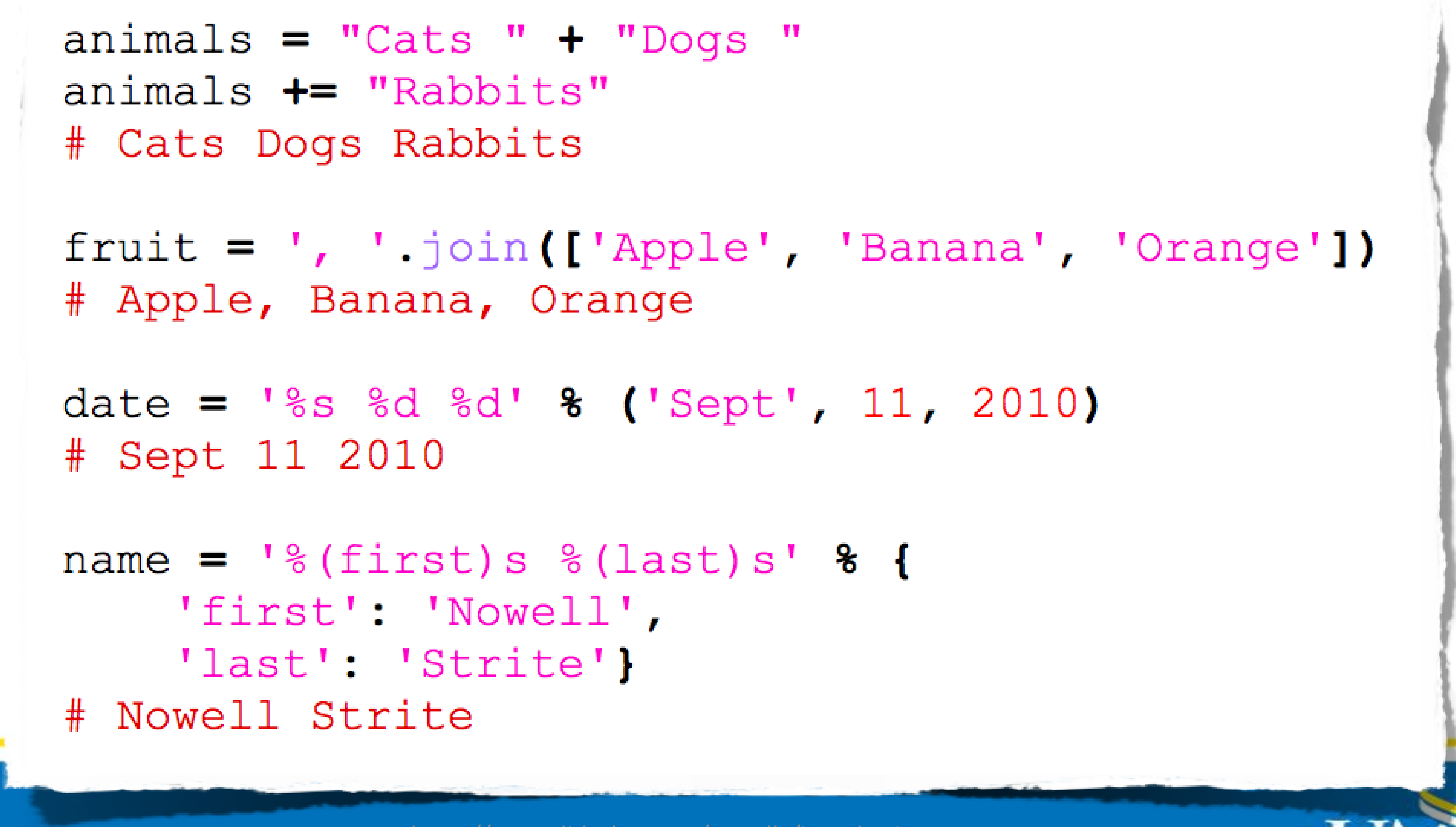
Basic String Operations

- we have variable astring = “Hello World!”
- print(len(astring)) => 12
- print(astring.index("o")) => 4
- print(astring.count("l")) => 3

Basic String Operations

- print(astring[3:7]) => lo w
- print(astring.startswith("Hello")) => true
- print(astring.endswith("asdfasdfasdf")) => false
- There is no function to reverse a string but we can do like this:
- print(astring[::-1]) => !dlrow olleH

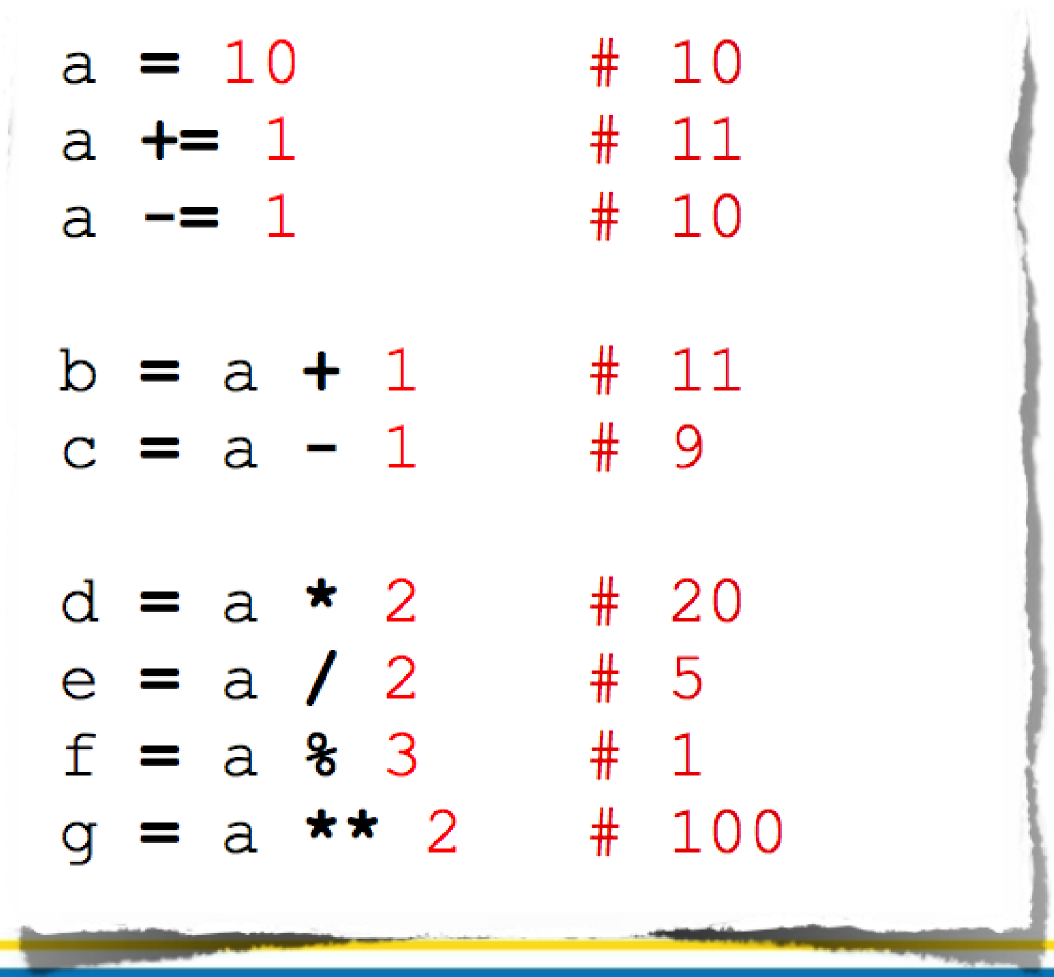
String Manipulation



```
animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010

name = '%(first)s %(last)s' % {
    'first': 'Nowell',
    'last': 'Strite'}
# Nowell Strite
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

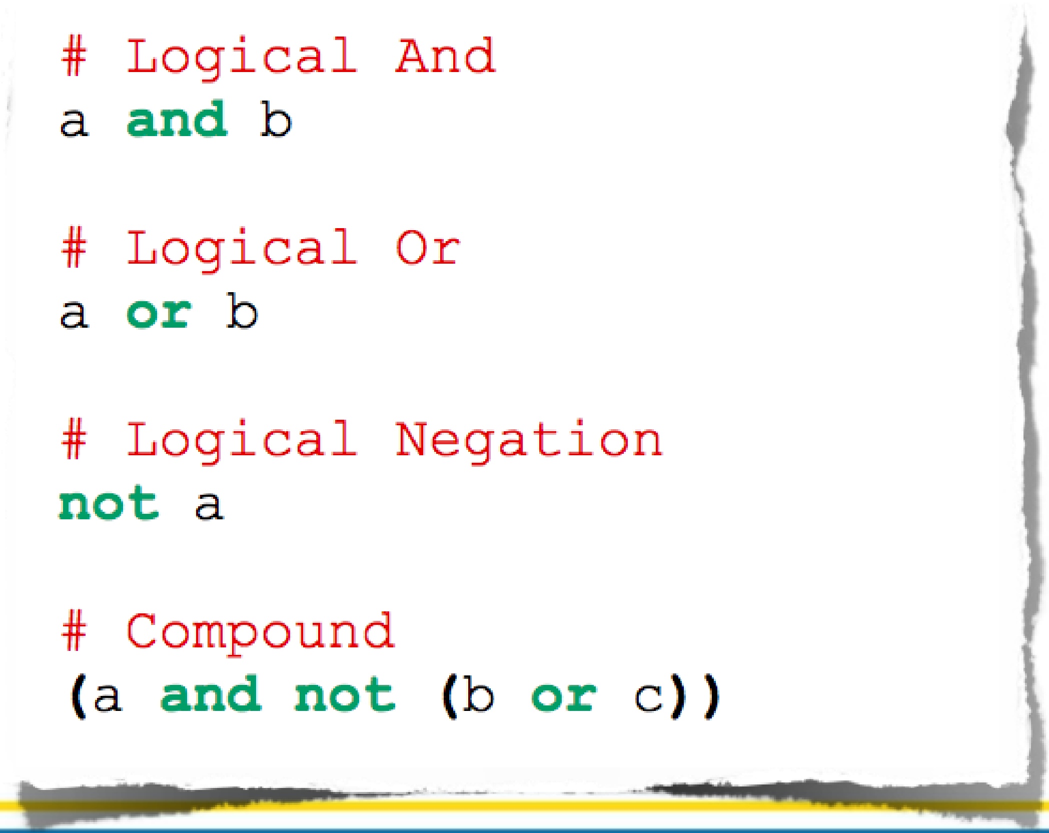
Arithmetic



```
a = 10              # 10
a += 1              # 11
a -= 1              # 10

b = a + 1           # 11
c = a - 1           # 9

d = a * 2           # 20
e = a / 2           # 5
f = a % 3           # 1
g = a ** 2          # 100
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

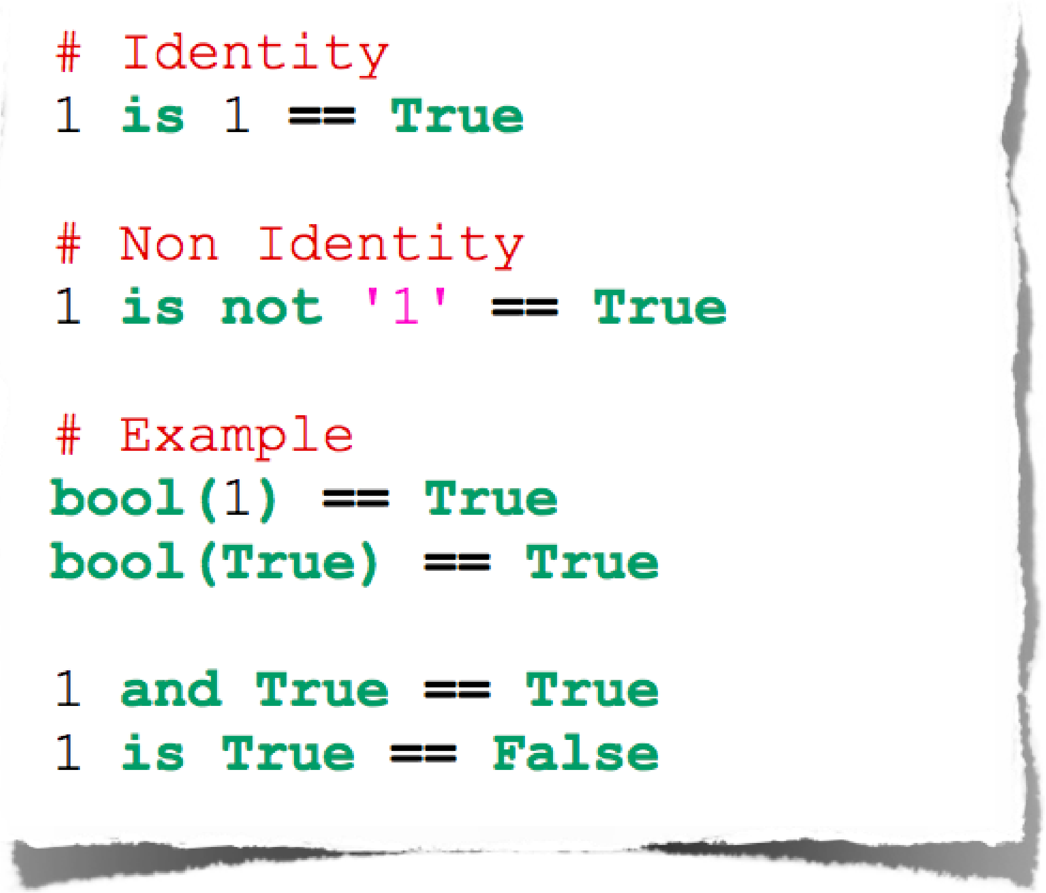
Logical Comparison



```
# Logical And
a and b

# Logical Or
a or b

# Logical Negation
not a

# Compound
(a and not (b or c))
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

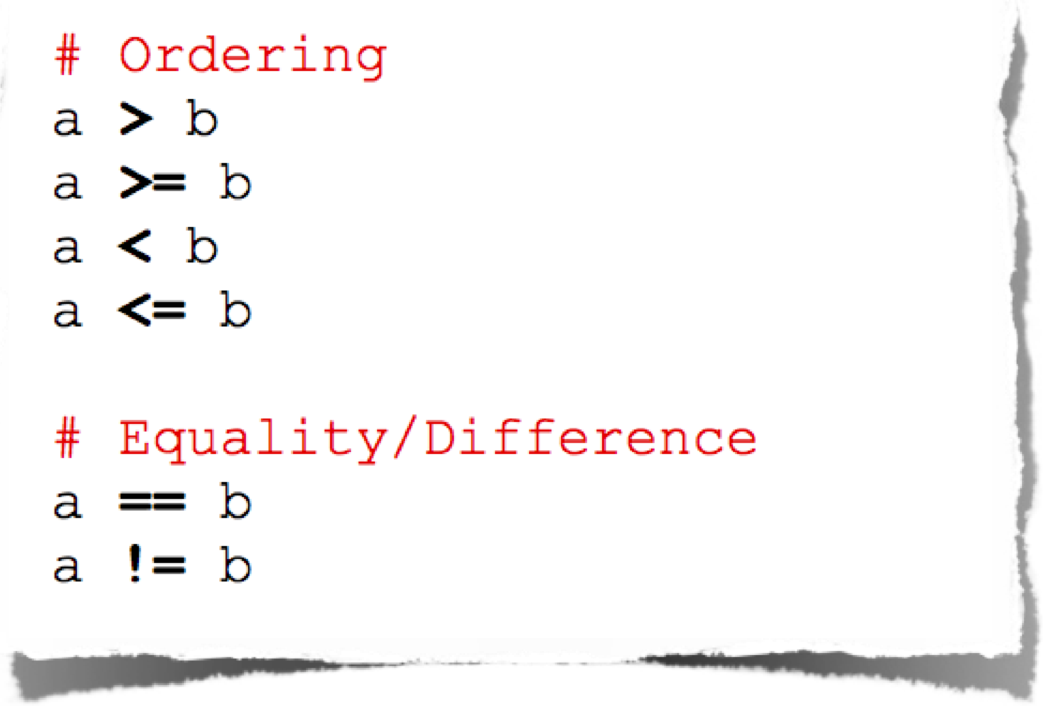
Identity Comparison



```
# Identity
1 is 1 == True

# Non Identity
1 is not '1' == True

# Example
bool(1) == True
bool(True) == True

1 and True == True
1 is True == False
```

https://www.slideshare.net/nowells/introduction-to-python-5182313

Arithmetic Comparison



```
# Ordering
a > b
a >= b
a < b
a <= b

# Equality/Difference
a == b
a != b
```

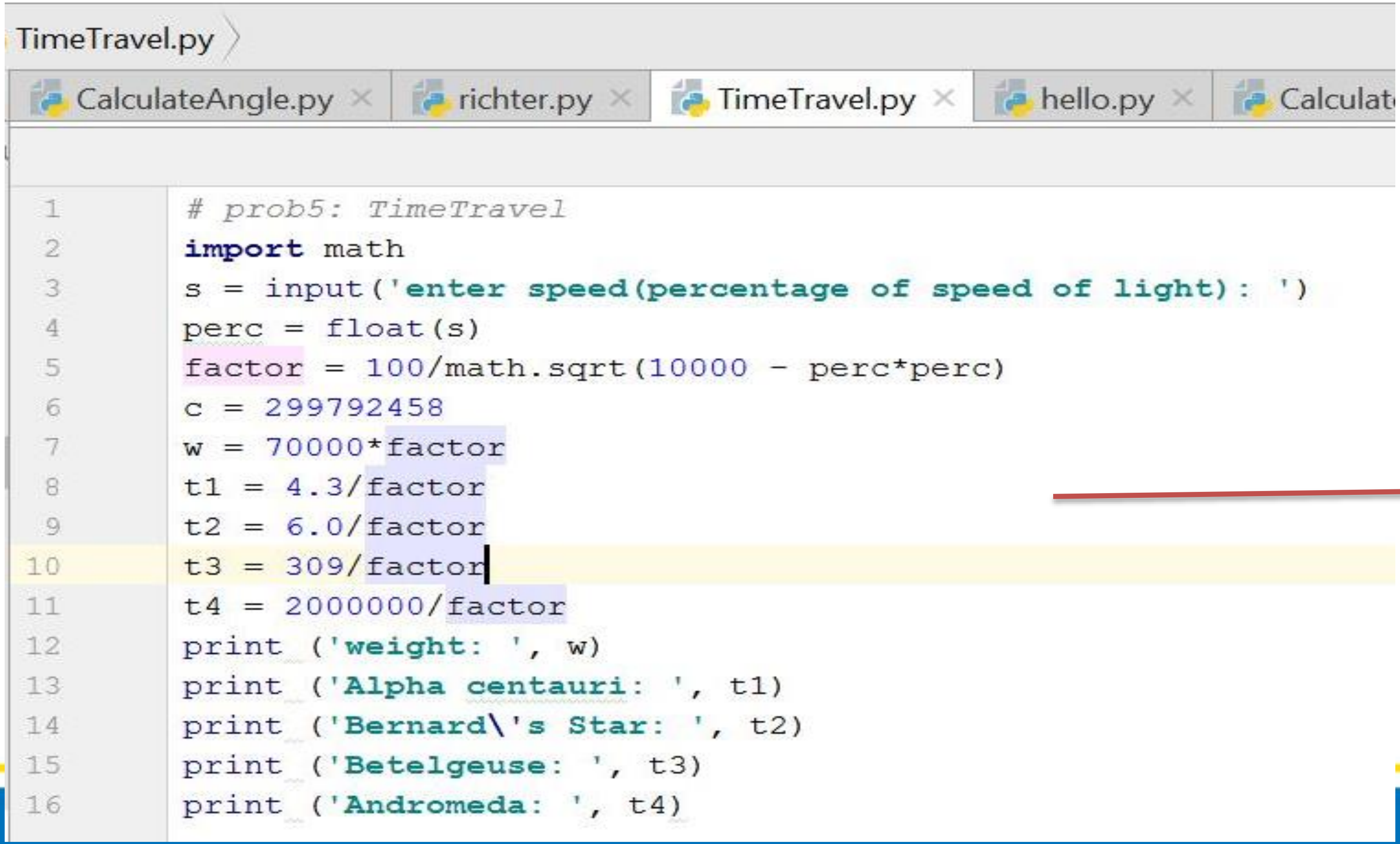
https://www.slideshare.net/nowells/introduction-to-python-5182313

Operators Precedence

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

The order in which operators are executed in any expression

Use case 3- Basic operations

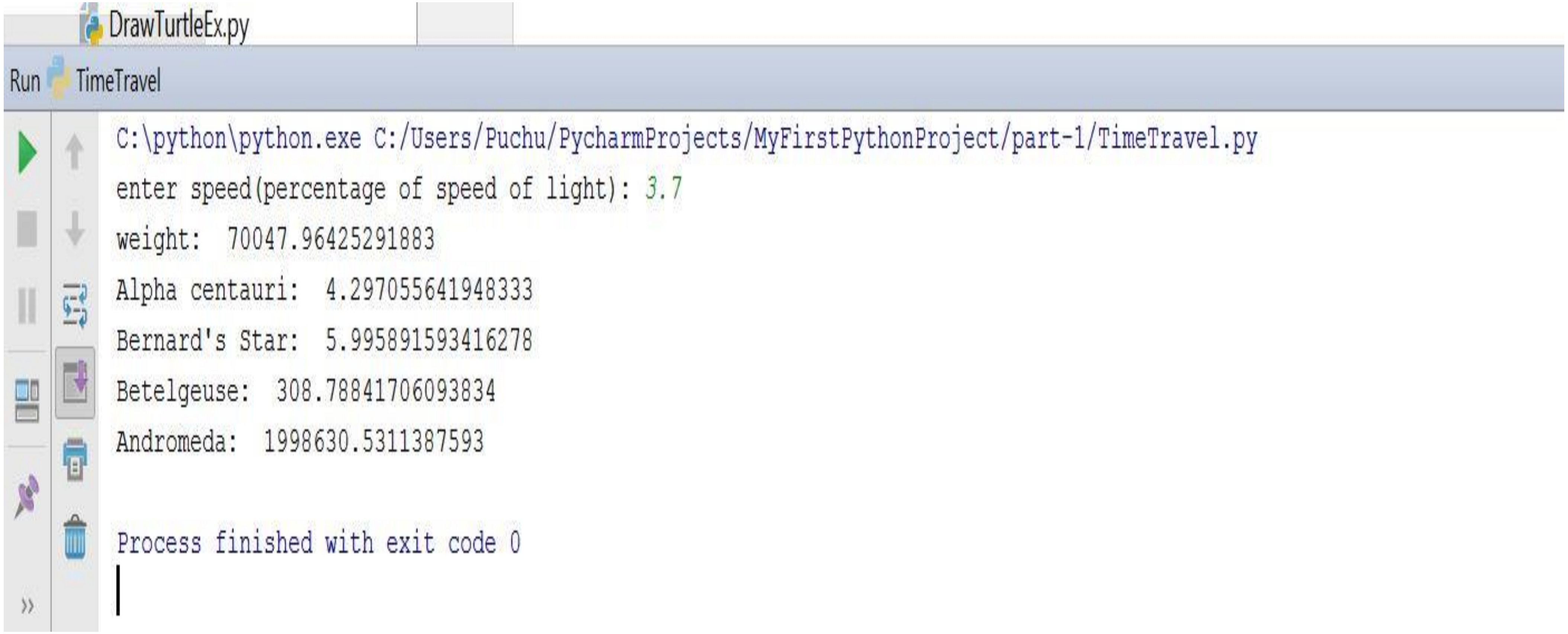


```python
# prob5: TimeTravel
import math
s = input('enter speed(percentage of speed of light): ')
perc = float(s)
factor = 100/math.sqrt(10000 - perc*perc)
c = 299792458
w = 70000*factor
t1 = 4.3/factor
t2 = 6.0/factor
t3 = 309/factor
t4 = 2000000/factor
print ('weight: ', w)
print ('Alpha centauri: ', t1)
print ('Bernard\'s Star: ', t2)
print ('Betelgeuse: ', t3)
print ('Andromeda: ', t4)
```

Basic Operations

https://github.com/galactocalypse/python

Use case 3- Output



```
C:\python\python.exe C:/Users/Puchu/PycharmProjects/MyFirstPythonProject/part-1/TimeTravel.py
enter speed(percentage of speed of light): 3.7
weight:  70047.96425291883
Alpha centauri:  4.297055641948333
Bernard's Star:  5.995891593416278
Betelgeuse:  308.78841706093834
Andromeda:  1998630.5311387593

Process finished with exit code 0
```

Importing and Modules

- Use classes & functions defined in another file to get additional functionality
- A Python module is a file with the same name (plus the *.py* extension)
- Like Java *import*, C++ *include*
- modules have private symbol tables
- Three formats of the command:

```
import somefile
from somefile import *
from somefile import className
```

- When a Python program starts it only has access to a basic functions and classes.

(“int”, “dict”, “len”, “sum”, “range”, ...)

import …

```
import somefile
```

- *Everything* in somefile.py gets imported.
- To refer to something in the file, append the text “somefile.” to the front of its name:

```
somefile.className.method(“abc”)
somefile.myFunction(34)
```

http://www.cs.brandeis.edu/~cs134/Python_tutorial.ppt

from … import …

```
from somefile import className
```

- Only the item *className* in somefile.py gets imported.
- After importing *className*, you can just use it without a module prefix. It's brought into the current namespace.

http://www.cs.brandeis.edu/~cs134/Python_tutorial.ppt

import the math module

```
>>> import math
>>> math.pi
3.1415926535897931
>>> math.cos(0)
1.0
>>> math.cos(math.pi)
-1.0
>>> dir(math)
['__doc__', '__file__', '__name__', '__package__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'hypot', 'isinf', 'isnan', 'ldexp', 'log', 'log10',
'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'trunc']
```

http://www.cs.brandeis.edu/~cs134/Python_tutorial.ppt

Basic Statements: The If Statement (1)

If statements have the following basic structure:

```
# inside the interpreter                      # inside a script
>>> if condition:        if condition:
...   action                 action
...
>>>
```

Subsequent indented lines are assumed to be part of the if statement. The same is true for most other types of python statements. A statement typed into an interpreter ends once an empty line is entered, and a statement in a script ends once an unindented line appears. The same is true for defining functions.

If statements can be combined with else if (elif) and else statements as follows:

```
if condition1:    # if condition1 is true, execute action1
  action1
elif condition2:  # if condition1 is not true, but condition2 is, execute
  action2         # action2
else:             # if neither condition1 nor condition2 is true, execute
  action3         # action3
```

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: The If Statement (2)

Conditions in if statements may be combined using and & or statements

```
if condition1 and condition2:
  action1
# if both  condition1 and condition2 are true, execute action1
if condition1 or condition2:
  action2
# if either condition1 or condition2 is true, execute action2
```

Conditions may be expressed using the following operations:

```
<, <=, >, >=, ==, !=, in
```

Somewhat unrealistic example:

```
>>> x = 2; y = 3; L = [0,1,2]
>>> if (1<x<=3 and 4>y>=2) or (1==1 or 0!=1) or 1 in L:
...   print 'Hello world'
...
Hello world
>>>
```

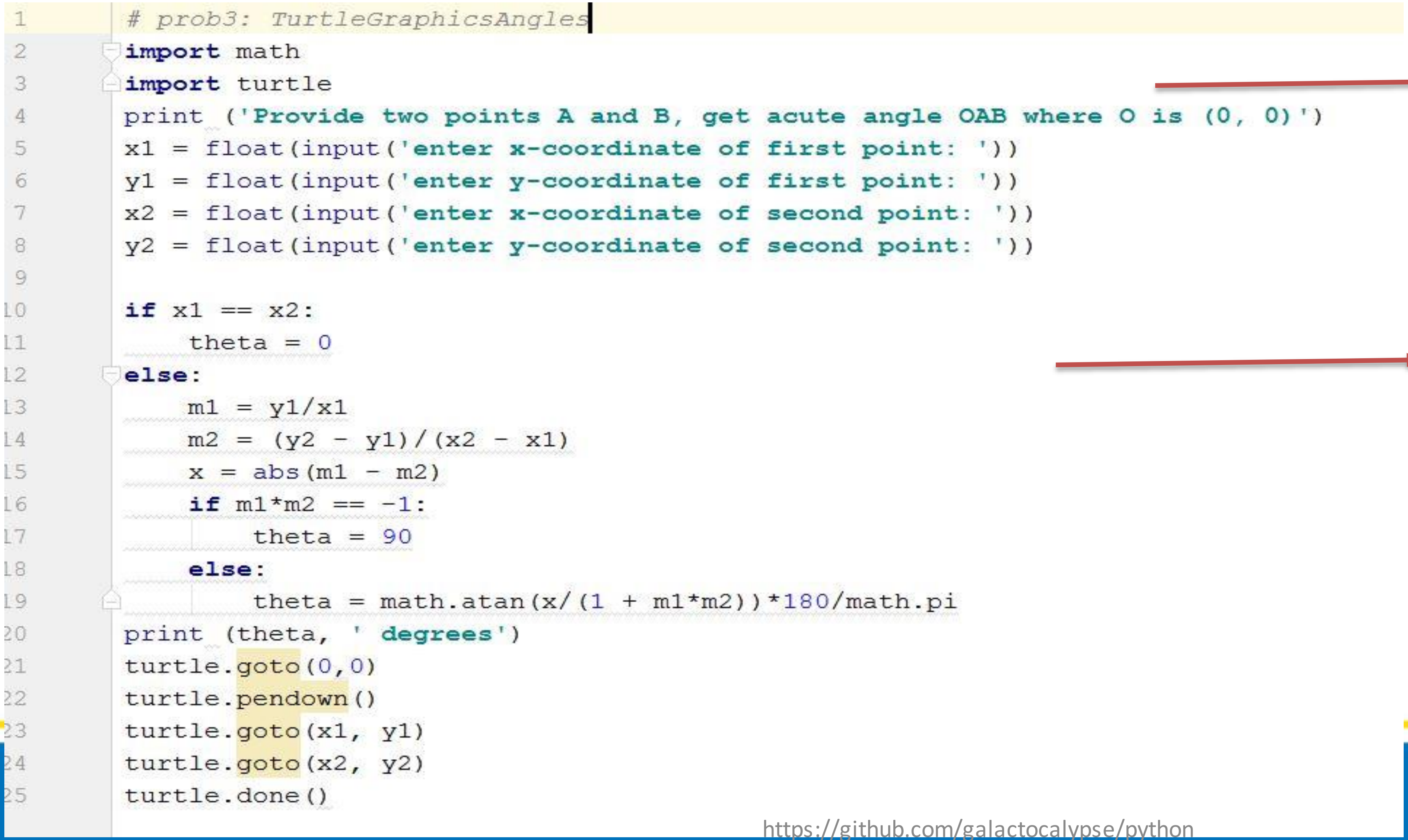
http://hifweb.lbl.gov/public/slides/python.ppt

If-Else-Statement examples

-
  ```
  if yearsWorked > 10 :
      bonus = 1000
  else :
      bonus = 500
  ```

-
  ```
  if age >= 65 :
      price = 0.85 * price
      numSeniors = numSeniors + 1
  else :
      nonSeniors = nonSeniors + 1
  ```

Use Case 4 – Import Turtle Graphics and create the two lines. Also calculate angle between them



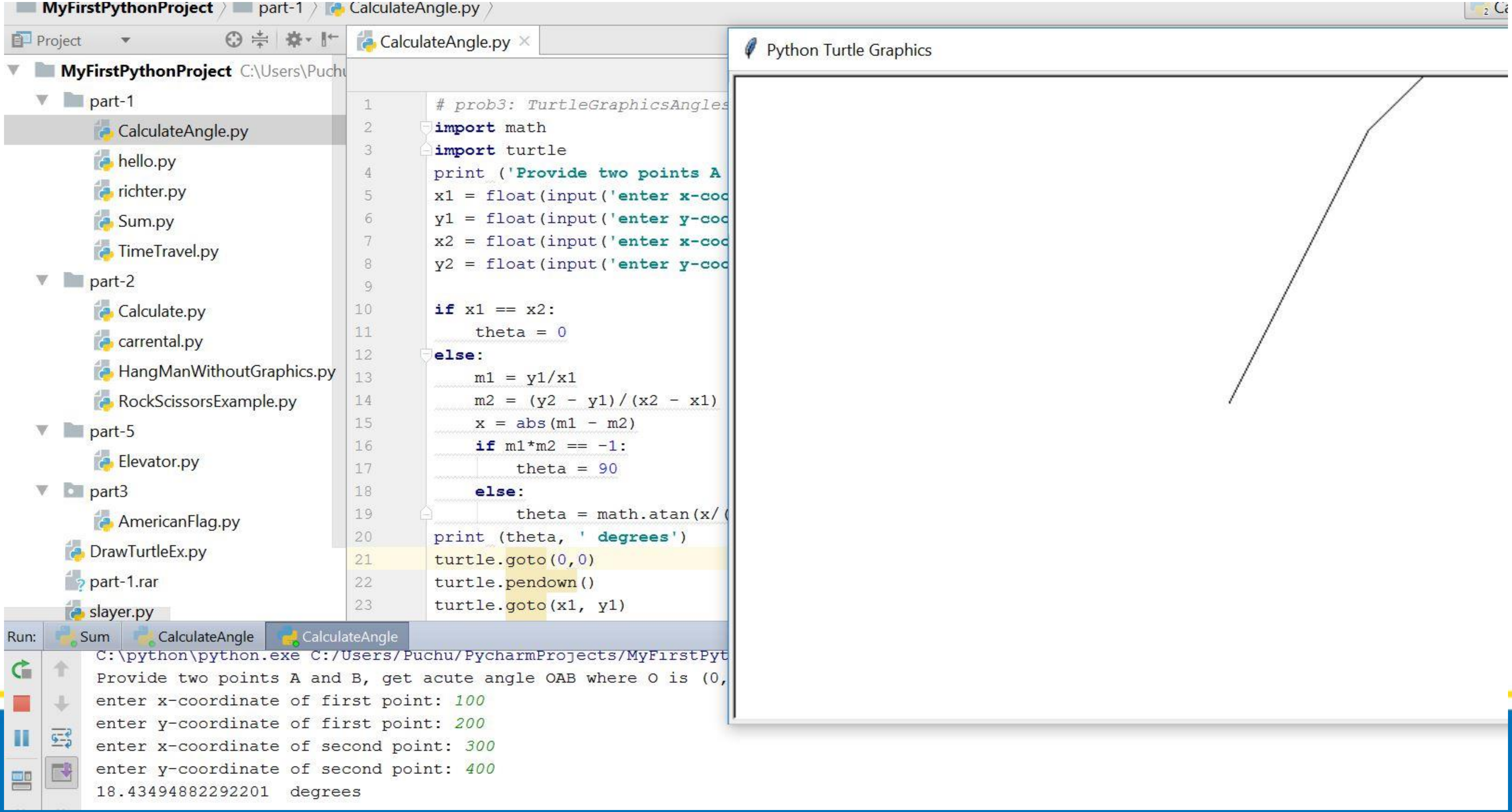
```python
# prob3: TurtleGraphicsAngles
import math
import turtle
print ('Provide two points A and B, get acute angle OAB where O is (0, 0)')
x1 = float(input('enter x-coordinate of first point: '))
y1 = float(input('enter y-coordinate of first point: '))
x2 = float(input('enter x-coordinate of second point: '))
y2 = float(input('enter y-coordinate of second point: '))

if x1 == x2:
    theta = 0
else:
    m1 = y1/x1
    m2 = (y2 - y1)/(x2 - x1)
    x = abs(m1 - m2)
    if m1*m2 == -1:
        theta = 90
    else:
        theta = math.atan(x/(1 + m1*m2))*180/math.pi
print (theta, ' degrees')
turtle.goto(0,0)
turtle.pendown()
turtle.goto(x1, y1)
turtle.goto(x2, y2)
turtle.done()
```

Import statements

If block

https://github.com/galactocalypse/python

Use Case 4 -Output



```python
# prob3: TurtleGraphicsAngles
import math
import turtle
print ('Provide two points A
x1 = float(input('enter x-coo
y1 = float(input('enter y-coo
x2 = float(input('enter x-coo
y2 = float(input('enter y-coo

if x1 == x2:
    theta = 0
else:
    m1 = y1/x1
    m2 = (y2 - y1)/(x2 - x1)
    x = abs(m1 - m2)
    if m1*m2 == -1:
        theta = 90
    else:
        theta = math.atan(x/(
print (theta, ' degrees')
turtle.goto(0,0)
turtle.pendown()
turtle.goto(x1, y1)
```

```
C:\python\python.exe C:/Users/Puchu/PycharmProjects/MyFirstPyt
Provide two points A and B, get acute angle OAB where O is (0,
enter x-coordinate of first point: 100
enter y-coordinate of first point: 200
enter x-coordinate of second point: 300
enter y-coordinate of second point: 400
18.43494882292201  degrees
```

Loops: break, continue, else

- break and continue like C
- else after loop exhaustion

```
for n in range(2,10):
  for x in range(2,n):
    if n % x == 0:
      print n, 'equals', x, '*', n/x
      break
  else:
    # loop fell through without finding a factor
    print n, 'is prime'
```

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: The While Statement (1)

While statements have the following basic structure:

```
# inside a script
while condition:
  action
```

As long as the condition is true, the while statement will execute the action

Example:

```
x = 1
while x < 4:  # as long as x < 4...
...   print x**2  # print the square of x
...   x = x+1     # increment x by +1
...
1               # only the squares of 1, 2, and 3 are printed, because
4               # once x = 4, the condition is false
9
>>>
```

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: The While Statement (2)

❑ Pitfall to avoid:

While statements are intended to be used with changing conditions. If the condition in a while statement does not change , the program will be stuck in an infinite loop until the user hits ctrl-C.

Example:

```
>>> x = 1
>>> while x == 1:
...   print 'Hello world'
...
```

Since x does not change, Python will continue to print “Hello world” until interrupted

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: The For Statement (1)

For statements have the following basic structure:

```
for item i in set s:
  action on item i
# item and set are not statements here; they are merely intended to clarify the relationships between i and s
```

Example:

```
>>> for i in range(1,7):
...   print i, i**2, i**3, i**4
...
1 1 1 1
2 4 8 16
3 9 27 81
4 16 64 256
5 25 125 625
6 36 216 1296
>>>
```

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: The For Statement (2)

The item i is often used to refer to an index in a list, tuple, or array

Example:

```
>>> L = [0,1,2,3]  # or, equivalently, range(4)
>>> for i in range(len(L)):
...   L[i] = L[i]**2
...
>>> L
[0,1,4,9]
>>>
```

Of course, we could accomplish this particular task more compactly using arrays:

```
>>> L = arange(4)
>>> L = L**2
>>> L
[0,1,4,9,]
```

http://hifweb.lbl.gov/public/slides/python.ppt

Range

- The range function specifies a range of integers:

`range(start, stop)` - the integers between start (inclusive) and stop (exclusive)

- It can also accept a third value specifying the change between values.

`range(start, stop, step)` - the integers between start (inclusive) and stop (exclusive) by step

http://hifweb.lbl.gov/public/slides/python.ppt

Basic Statements: Combining Statements

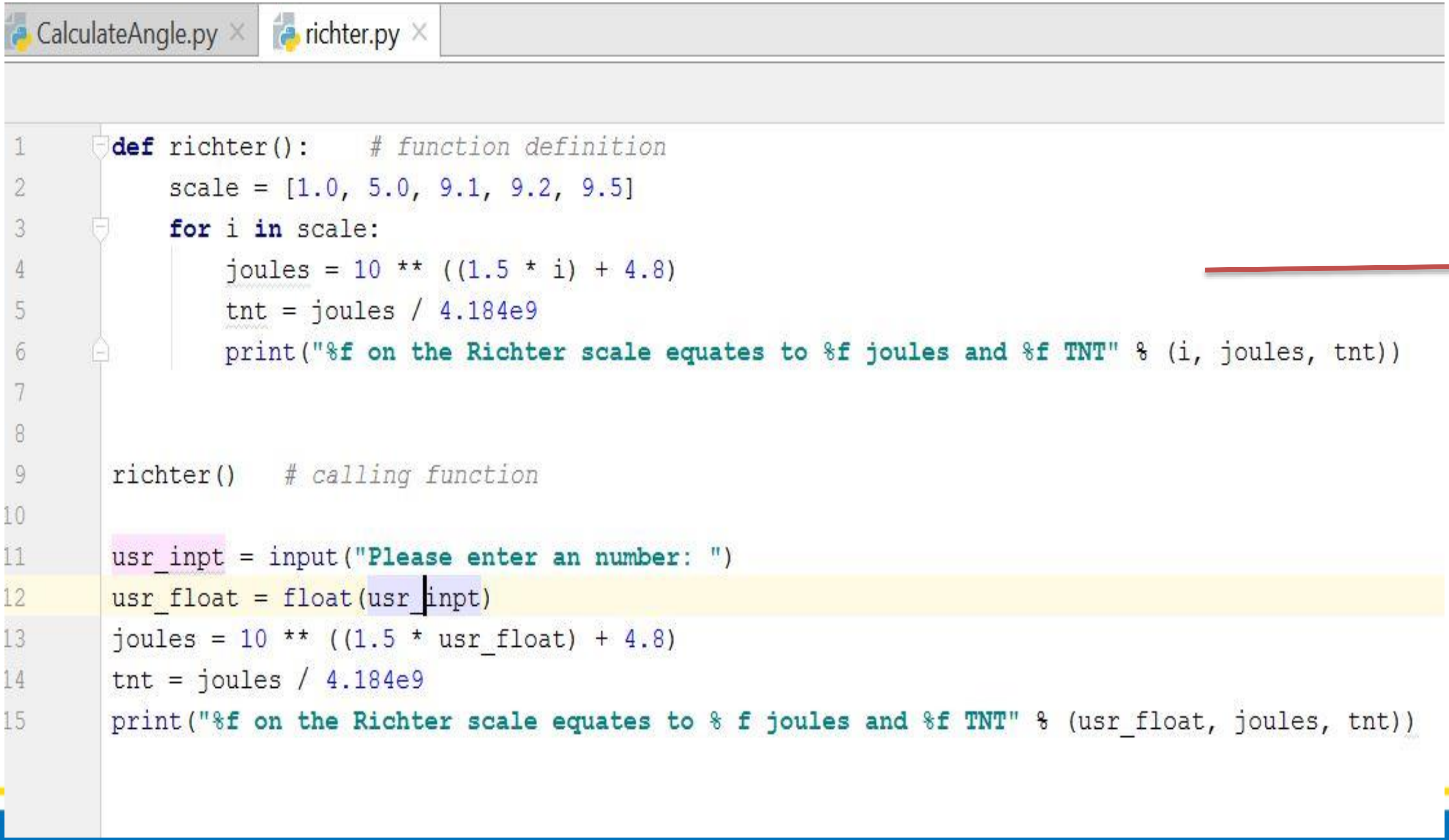
The user may combine statements in a myriad of ways

Example:

```
>>> L = [0,1,2,3]   # or, equivalently, range(4)
>>> for i in range(len(L)):
...   j = i/2.
...   if j – int(j) == 0.0:
...     L[i] = L[i]+1
...   else: L[i] = -i**2
...
>>> L
[1,-1,3,-9]
>>>
```

http://hifweb.lbl.gov/public/slides/python.ppt

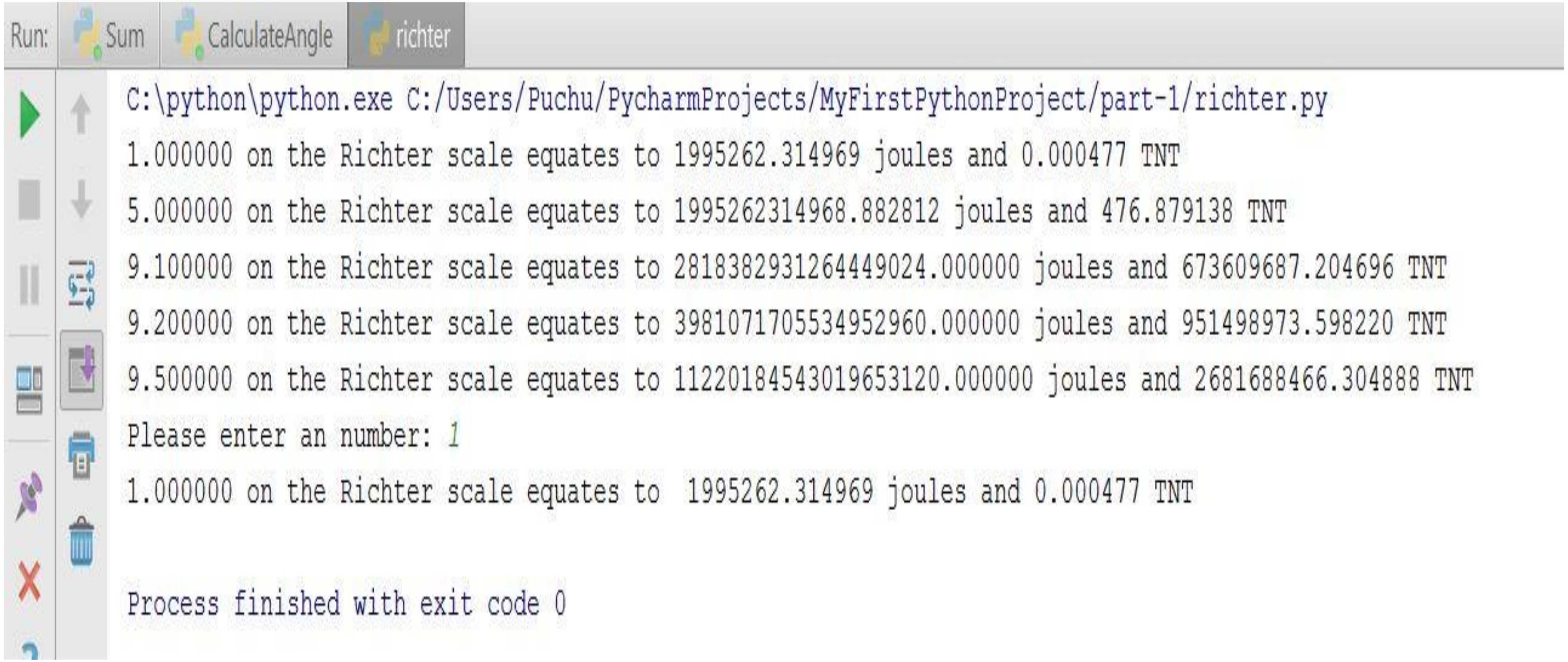
Use Case 5 – Basic for loop



```
def richter():     # function definition
    scale = [1.0, 5.0, 9.1, 9.2, 9.5]
    for i in scale:
        joules = 10 ** ((1.5 * i) + 4.8)
        tnt = joules / 4.184e9
        print("%f on the Richter scale equates to %f joules and %f TNT" % (i, joules, tnt))


richter()    # calling function

usr_inpt = input("Please enter an number: ")
usr_float = float(usr_inpt)
joules = 10 ** ((1.5 * usr_float) + 4.8)
tnt = joules / 4.184e9
print("%f on the Richter scale equates to % f joules and %f TNT" % (usr_float, joules, tnt))
```

For loop

Use Case 5 – Output



```
C:\python\python.exe C:/Users/Puchu/PycharmProjects/MyFirstPythonProject/part-1/richter.py
1.000000 on the Richter scale equates to 1995262.314969 joules and 0.000477 TNT
5.000000 on the Richter scale equates to 1995262314968.882812 joules and 476.879138 TNT
9.100000 on the Richter scale equates to 2818382931264449024.000000 joules and 673609687.204696 TNT
9.200000 on the Richter scale equates to 3981071705534952960.000000 joules and 951498973.598220 TNT
9.500000 on the Richter scale equates to 11220184543019653120.000000 joules and 2681688466.304888 TNT
Please enter an number: 1
1.000000 on the Richter scale equates to  1995262.314969 joules and 0.000477 TNT


Process finished with exit code 0
```

References

- https://www.slideshare.net/nowells/introduction-to-python-5182313
- https://www.slideshare.net/sujithkumar9212301/introduction-to-python-36647807
- https://github.com/galactocalypse/python
- http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/download-course-materials/