Programming and Simulation

Amy Lien

Goddard Space Flight Center

ASTR 288C, Lecture 6, 2017/10/09

ASTRONOMER

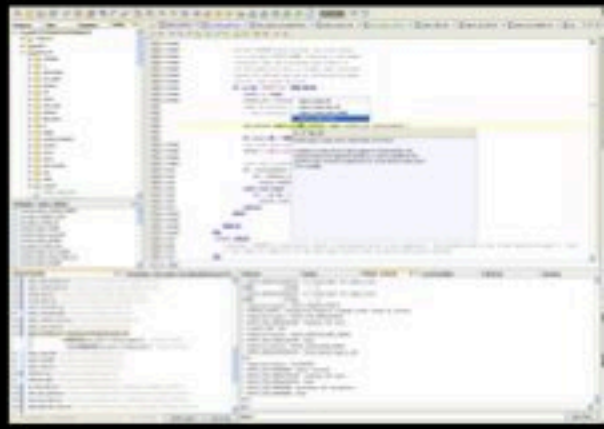
What my friends think I do

What my mom thinks I do

What society thinks I do

What the university thinks I do

What I think I do



What I really do

Why python?

Oct 2017	Oct 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.431%	-6.37%
2	2		C	8.374%	-1.46%
3	3		C++	5.007%	-0.79%
4	4		C#	3.858%	-0.51%
5	5		Python	3.803%	+0.03%
6	6		JavaScript	3.010%	+0.26%
7	7		PHP	2.790%	+0.05%
8	8		Visual Basic .NET	2.735%	+0.08%
9	11	^	Assembly language	2.374%	+0.14%
10	13	^	Ruby	2.324%	+0.32%
11	15	^^	Delphi/Object Pascal	2.180%	+0.31%
12	9	v	Perl	1.963%	-0.53%
13	19	^^	MATLAB	1.880%	+0.26%
14	23	^^	Scratch	1.819%	+0.69%
15	18	^	R	1.684%	-0.06%

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

Why python?

- Many people are using it, not just astronomers
- Easy to start
- (A lot) more human friendly than the basic language like C
- Powerful (combine coding and plotting)
- Many supporting packages for astronomy research.

Why python?

- Many people are using it, not just astronomers
- Easy to start
- (A lot) more human friendly than basic languages like C
- Powerful (combine coding and plotting)
- Many supporting packages for astronomy research.
- **IT'S FREE!**

Learning python

- Need to think like a computer – simple logic
- Practice, practice, practice
- Google is extremely useful
- Lots of good online resources, here are a few:
  - https://www.python.org/
  - https://www.learnpython.org/
  - http://docs.python-guide.org/en/latest/intro/learning/

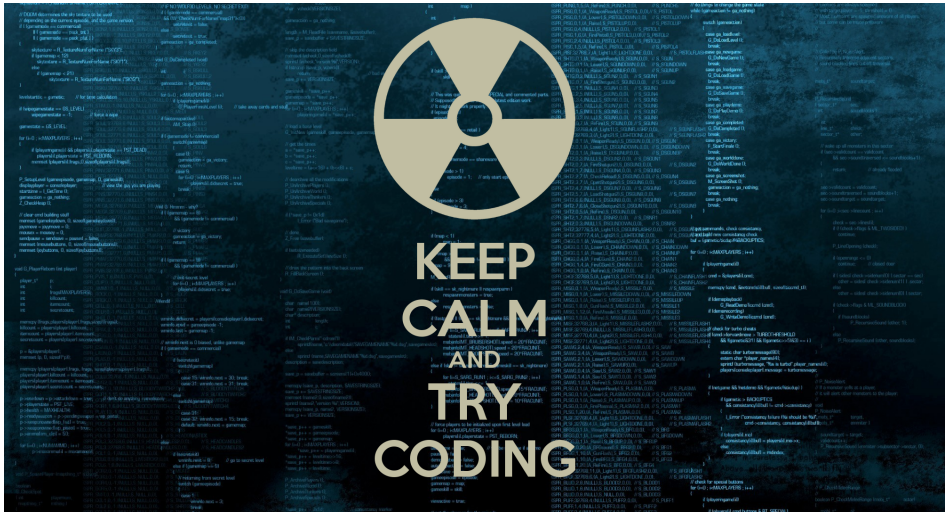
A few note for python class

- You WILL run into problems in your code.
- If you run into too many problems and can't finish today's lab, don't panic. Just let me know and you can continue to work from home. I am happy to help you finish it after today's class and during office hour.
- START THIS WEEK'S HOMEWORK EARLY!!
- Office hour:

  Friday, 4:30 to 5:30 pm

  (or by appointment)

  at ATL 0251A.



Outline
Python I and II

- Basic structure
  - Print “Hello World!”
- For loop
- If statement
- Input and output (open files, save files)
- Logical elements (and, or, ==)
- Array (1d, 2d)
- Plot
- Managing FITS file in python

Next week

You can find the example python codes at /n/ursa/A288C/alien/python_template

Basic structure

- Code need to have name *.py
- After finishing editing the filename.py file, run the code by typing

  python filename.py
- Print “Hello World!”
  - Example code: Hello_world.py

```
print "Hello World!"
```

  - Output:

```
ursa% python Hello_world.py
Hello World!
```

for loop

- Example code: for_loop.py

```
## An example of using for loop for print 1 to 10
for i in range(1,11):
        print i
```

- Output:

```
ursa% python for_loop.py
1
2
3
4
5
6
7
8
9
10
```

for loop

- Example code: for_loop.py

```
## An example of using for loop for print 1 to 10
for i in range(1,11):
        print i
```

The end is exclusive

- Output:

```
ursa% python for_loop.py
1
2
3
4
5
6
7
8
9
10
```

for loop

- Example code: for_loop.py

```
## An example of using for loop for print 1 to 10
for i in range(1,11):
    print i
```

Remember the tab!

```
1
2
3
4
5
6
7
8
9
10
```

for loop

- Example code: for_loop_sum.py

```
## An example of using for loop to add up 1 to 10

## set the initial value to the number so we can sum it up later
i_tot = 0
## Go through 1 to 10 and add it up
for i in range(1,11):
        ## Set the new i_tot equals to the old i_tot plus i
        i_tot = i_tot + i
        ## print out i_tot for each step to see the progress
        print i, i_tot

## print out the final number
print 'Total = ', i_tot
```

- Output:

```
ursa% python for_loop_sum.py
1 1
2 3
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
Total =  55
```

for loop

- Example code: for_loop_sum.py

```
## An example of using for loop to add up 1 to 10

## set the initial value to the number so we can sum it up later
i_tot = 0
## Go through 1 to 10 and add it up
for i in range(1,11):
        ## Set the new i_tot equals to the old i_tot plus i
        i_tot = i_tot + i
        ## print out i_tot for each step to see the progress
        print i, i_tot

## print out the final number
print 'Total = ', i_tot
```

Or i_tot += i

- Output:

```
ursa% python for_loop_sum.py
1 1
2 3
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
Total =  55
```

1-d array

- Example code: array_1d.py

```
## This example creates an 1-d array using for loop

## Declare an empty array
test_array_1d = []

## Go through a for loop to append (fill out) number in the array
for i in range(0,11):
        test_array_1d.append(i)

## Print out the array
print test_array_1d

## Print out the 5th element of the array
print '5th element:', test_array_1d[5]

## Print out the length of the array
print 'Length of the array:', len(test_array_1d)

## We can also use a for loop to get each element of the array.
## This for loop go through the length of the array.
for i in range(len(test_array_1d)):
        print test_array_1d[i]
```

1-d array

- Example code: array_1d.py

```
## This example creates an 1-d array using for loop

## Declare an empty array
test_array_1d = []

## Go through a for loop to append (fill out) number in the array
for i in range(0,11):
        test_array_1d.append(i)

## Print out the array
print test_array_1d

## Print out the 5th element of the array
print '5th element:', test_array_1d[5]

## Print out the length of the array
print 'Length of the array:', len(test_array_1d)

## We can also use a for loop to get each element of the array.
## This for loop go through the length of the array.
for i in range(len(test_array_1d)):
        print test_array_1d[i]
```

Remember python array element starts from 0.

if statement

- Example code: if_statement.py

```
## An example of using if statement
## Going throught the for loop from 1 to 10,
## only print out values larger or equal to 5.
print "Values above or equal to 5"
for i in range(1,11):
        if (i >= 5):
                print i

## print out values below 2 and above 7
print "Values smaller than 2 and larger than 7"
for i in range(1,11):
        if (i < 2 or i > 7):
                print i
```

- Output

```
Values above or equal to 5
5
6
7
8
9
10
Values smaller than 2 and larger than 7
1
8
9
10
```

if else statement

- Example code: if_else_statement.py

```
## An example of using if-else statement
## Going throught the for loop from 1 to 10,
## print 'small' for values smaller than 5
## and print 'large' for values larger or equal to 5.
for i in range(1,11):
        if (i >= 5):
                print i, 'large'
        else:
                print i, 'small'
```

- Output

```
ursa% python if_else_statement.py
1 small
2 small
3 small
4 small
5 large
6 large
7 large
8 large
9 large
10 large
```

Math

- Example code: math_example.py

```
## Import relevant package
import math

## Set up variables x and y
x = 5
y = 7

## Print out the sum of x and y
print 'x+y = ', (x+y)

## Alternatively, you can do it like this
z = x + y
print 'z = ', z
```

- Output

```
x+y =  12
z =  12
```

Input
- read in a file

- Example code: read_file.py

```
import sys
import math

## Open the file to read
f_input = open('summary_general.txt','r')

## Using a for loop to go through each line in the file
for line in f_input.readlines():
        print line

## close file
f_input.close
```

Usually you want to import at least there two packages

Input
- read the value of each column in file

- Example code: read_column_in_file.py

```
import sys
import math

## Open the file to read
f_input = open('summary_general.txt','r')

## Using a for loop to go through each line in the file
for line in f_input.readlines():
        ## Skip the comment lines (which starts with the '#' symbol).
        if '#' not in line[0]:
                ## Make column based on the '|' separation
                column = line.split('|')
                ## Give the interested column element an easily understandable name
                GRBname = column[0]
                Trig_ID = column[1]

                ## Print out GRBname and Trigger ID
                print GRBname, Trig_ID


## Close file
f_input.close
```

## GRBname	Trig_ID	Trig_time_met
GRB170912B	772052	526890846.512
GRB170912A	772006	526872876.608

Basic variable types

Function	Converting what to what	Example

`int()`	string, floating point → integer	`>>> int('2014')`
		`2014`
		`>>> int(3.141592)`
		`3`

`float()`	string, integer → floating point number	`>>> float('1.99')`
		`1.99`
		`>>> float(5)`
		`5.0`

`str()`	integer, float, list, tuple, dictionary → string	`>>> str(3.141592)`
		`'3.141592'`
		`>>> str([1,2,3,4])`
		`'[1, 2, 3, 4]'`

Source: http://www.pitt.edu/~naraehan/python2/data_types_conversion.html

Convert variable types

- Example code: convert_variable_type.py

```
import sys
import math

## Open the file to read
f_input = open('summary_general.txt','r')

## Using a for loop to go through each line in the file
for line in f_input.readlines():
        ## Skip the comment lines (which starts with the '#' symbol).
        if '#' not in line[0]:
                ## Make column based on the '|' separation
                column = line.split('|')
                ## Give the interested column element an easily understandable name
                GRBname = column[0]
                Trig_ID = column[1]

                ## Things read in from a file are in the "string" format,
                ## to do any math of the value, we need to convert them to float first.

                ## Make sure there are no 'N/A' in Trig_ID before we convert it to float
                if ('N/A' not in Trig_ID):
                        Trig_ID_float = float(Trig_ID)

                        ## Now we can to math of Trig_ID
                        Trig_ID_new = Trig_ID_float + 0.5

                        ## Print out the old and new Trig ID
                        print Trig_ID, Trig_ID_new
```

Save output to a file

- Example code: save_file.py

```
import sys
import math

## Open the file to read
f_input = open('summary_general.txt','r')

## Open an empty file to save the output
f_output = open('test_output.txt','w')

## Using a for loop to go through each line in the file
for line in f_input.readlines():
        ## Skip the comment lines (which starts with the '#' symbol).
        if '#' not in line[0]:
                ## Make column based on the '|' separation
                column = line.split('|')
                ## Give the interested column element an easily understandable name
                GRBname = column[0]
                Trig_ID = column[1]

                ## Things read in from a file are in the "string" format,
                ## to do any math of the value, we need to convert them to float first.

                ## Make sure there are no 'N/A' in Trig_ID before we convert it to float
                if ('N/A' not in Trig_ID):
                        Trig_ID_float = float(Trig_ID)

                        ## Now we can to math of Trig_ID
                        Trig_ID_new = Trig_ID_float + 0.5

                        ## Print out the old and new Trig ID
                        print Trig_ID, Trig_ID_new

                        ## Now save the old and new Trig ID to the output file
                        ## The file can only save things in the "string" format,
                        ## so need to convert float back into string.
                        f_output.write(Trig_ID + ' ' + str(Trig_ID_new) + '\n')
                        ## '\n' is the line break


## Close file
f_input.close
f_output.close
```

Some useful python syntax

Operation	Result	Notes
`x or y`	if *x* is false, then *y*, else *x*	(1)
`x and y`	if *x* is false, then *x*, else *y*	(2)
`not x`	if *x* is false, then `True`, else `False`	(3)

Operation	Meaning	Notes
`<`	strictly less than	
`<=`	less than or equal	
`>`	strictly greater than	
`>=`	greater than or equal	
`==`	equal	
`!=`	not equal	(1)
`is`	object identity	
`is not`	negated object identity	

Source: https://docs.python.org/2/library/stdtypes.html

Some useful python math syntax

Operation	Result	Notes
`x + y`	sum of *x* and *y*	
`x - y`	difference of *x* and *y*	
`x * y`	product of *x* and *y*	
`x / y`	quotient of *x* and *y*	(1)
`x // y`	(floored) quotient of *x* and *y*	(4)(5)
`x % y`	remainder of `x / y`	(4)
`-x`	*x* negated	
`+x`	*x* unchanged	
`abs(x)`	absolute value or magnitude of *x*	(3)
`int(x)`	*x* converted to integer	(2)
`long(x)`	*x* converted to long integer	(2)
`float(x)`	*x* converted to floating point	(6)
`complex(re,im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.	
`c.conjugate()`	conjugate of the complex number *c*. (Identity on real numbers)	
`divmod(x, y)`	the pair `(x // y, x % y)`	(3)(4)
`pow(x, y)`	*x* to the power *y*	(3)(7)
`x ** y`	*x* to the power *y*	(7)

Source: https://docs.python.org/2/library/stdtypes.html

Some useful debugging trick

- sys.exit(0) - end program here
  - This make you able to run only part of the code to figure out which part is causing problems.
- print
  - Print out the value produced by each step
  - Check if these values are the expected ones.