Python Programming for Novice

Malvika Sharan & Olav Vahtras

Workshop: Software Writing Skills for Young Researchers
2015.09.23 - 2015.09.25

GFZ Helmholtz-Zentrum Potsdam, Germany

Topics to be discussed

- Basics (What is? And How to?)
- How to read, write, manipulate and process information
- How to get the repeated task done without repeated coding
- Most importantly, using Python to deal with your data
- Link to all the documents for this course
  https://github.com/malvikasharan/software_writing_skills_potsdam/tree/potsdam
- References for this course: 'A Byte of Python' by Swaroop C. H. along with several resources mentioned in the GitHub repository
- ASK when something is not clear

Python Programming for Novice

Malvika Sharan & Olav Vahtras

Day – 1

2015-09-23

“Hello World!”

1. Open your terminal
2. Run Python (Type ‘python’ + Enter-key)
   - You would see information similar to this:

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

3. Type quit() + Enter-key to quit
4. Traditional “Hello World!”
   - Run Python
   - Type and enter+key
     - print “Hello World!”

“Hello World!”

1. Open your terminal
2. Run Python (Type ‘Python’ + Enter-key)
   - You would see something similar like this information

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

3. Type quit() + Enter-key to quit
4. Traditional “Hello World!”

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>>
```

Literal Constants

- We use the values of *literals* literally
  - These values represent themselves and nothing else
- int(): Integers or whole numbers of any length like 2, 30, 1000
- float(): Floats or floating point numbers like 1.9, 30.01, 1e3
- str(): String or sequence of characters like “Hello”, “1 World”, “&”
  - Single quote and double quotes
    - Either of the quote can be used to print a string
    - Spaces, tabs, comma etc. inside the quotes are preserved
  - Triple Quotes
    - Multi line strings can be specified by triple quotes (‘’’ or “””)

Literal Constants

- you use the values of *literals* literally
  - These values represent themselves and nothing else
- int(): Integers or whole numbers of any length like 2, 30, 1000
- float(): Floats or floating point numbers like 1.99, 30.001, 1e3
- str(): String or sequence of characters like “Hello World”
  - Single quote and double quotes
    - Either of the quote can be used to print a string
    - Spaces, tabs, comma etc. inside the quotes are preserved
  - Triple Quotes
    - Multi line strings can be specified by triple quotes (‘’’ or “””)

```
>>> print '''This is amazing,
... because I can press enter,
... and keep writing without any error'''
This is amazing,
because I can press enter,
and keep writing without any error
```

Literal Constants

- you use the values of *literals* literally
  - These values represent themselves and nothing else
- int(): Integers or whole numbers of any length like 2, 30, 1000
- float(): Floats or floating point numbers like 1.99, 30.001, 1e3
- str(): String or sequence of characters like “Hello World”
  - Single quote and double quotes
    - Either of the quote can be used to print a string
    - Spaces, tabs, comma etc. inside the quotes are preserved
  - Triple Quotes
    - Multi line strings can be specified by triple quotes (‘’’ or “””)

```
>>> print '''This is amazing,
... because I can press enter,
... and keep writing without any error'''
This is amazing,
because I can press enter,
and keep writing without any error
```

```
>>> print "this is amazing,
  File "<stdin>", line 1
    print "this is amazing,
                           ^
SyntaxError: EOL while scanning string literal
>>>
```

“Hello World!” and Literal Constants

Exercises:

1. Print more strings
2. Print numbers
   - Hint: does not need quotes
3. Print a float with int() and integer with float() types
4. Print multiple strings and numbers together
   - Hint 1: use comma as separator
   - Hint 2: use plus sign (+) to connect
     - gives error (or sum, when int1+int2) with numbers, how to solve this?
5. Print a string with double quote
   - Example: “this string appears with double quotes”

Variables

- Using literal constants will become boring
  - What does Python do if I am writing everything on my own?
- *Variables* are variables, their values can change
  - Allow storing any information in your computer's memory
  - We need a method to access them by giving them names (Identifiers)

Rules:

- The identifiers are case sensitive
- myname and MyName are different
- The name starts with a letter or alphabet
- Followed by letters, underscores or digits (name_1)

Variables

- Using literal constants will become boring
  - "What would Python do if I am writing everything on my own?"
- *Variables* are variables, their values can change
  - Allow storing any information in your computer's memory
  - We need a method to access them by giving them names (Identifiers)

Rules:

- The identifiers are case sensitive
- myname and MyName are different
- The name starts with a letter or alphabet
- Followed by letters, underscores or digits

```
>>> name = 'John Doe'
>>> print "Name is", name
Name is John Doe
>>> age = 45
>>> print "Age is", age
Age is 45
>>> print "Age of", name, "is", age
Age of John Doe is 45
>>>
```

String Formatting

- The string format operators allows formatting output of an ordered strings
  - Previous example: print “Age of”, name, “is”, age
  - Smarter way 1: print “Age of {} is {}”.format(name, age)
  - Smarter way 2: print “Age of %s is %s” % (name, age)
    - There are several options with %:
    - %s: string
    - %d: decimal point number
    - %f: float point number
    - %.2f: float point number to 2 decimal places
    - Etc.

```
>>> name = 'John Doe'
>>> print "Name is", name
Name is John Doe
>>> age = 45
>>> print "Age is", age
Age is 45
>>> print "Age of", name, "is", age
Age of John Doe is 45
>>>
```

Print the output by string format operators

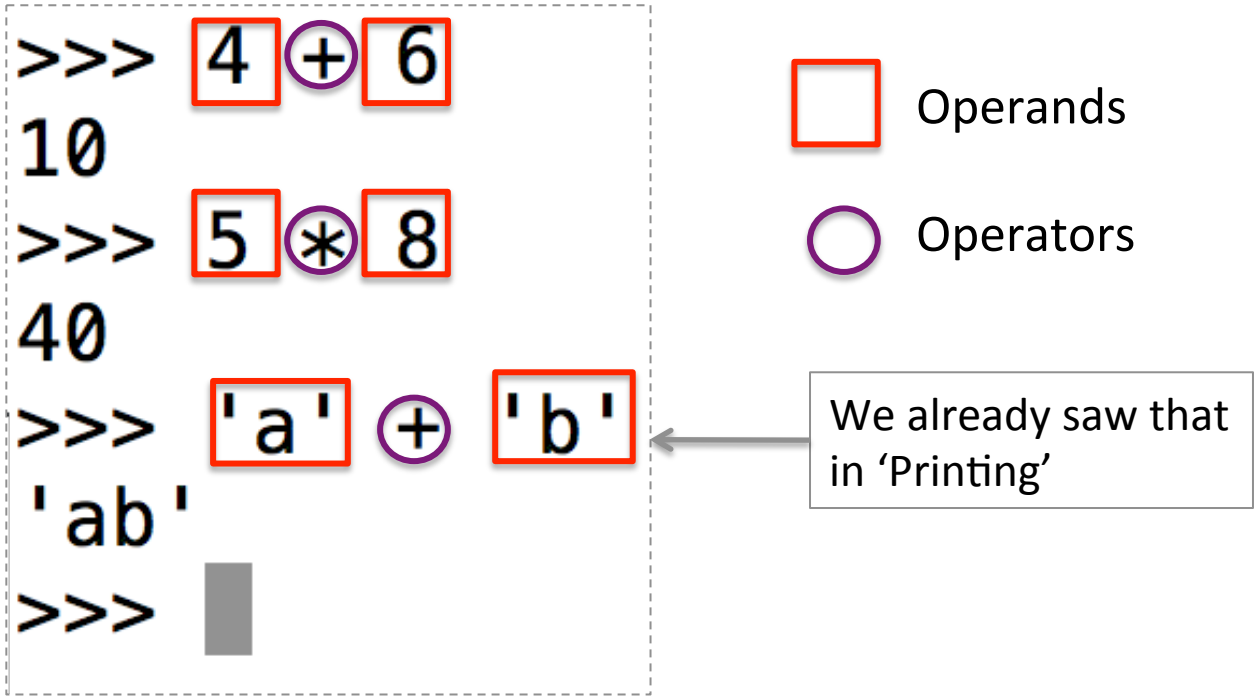
Variables Exercises

Exercises:

1. Assign any value to variable i, print and check the value
   - Example: i = 10
2. Reassign a different value to i, print and check
   - Example: i = 23
   - You just overwrote the value of i, the last assigned value is the current value
3. Assign multiple variables same values, print and check
   - Example: a = b = c = 59
4. Assign multiple variables (a, b, c) different values, print and check
   - In separate lines (a = 1 + Enter-key, b = 2 + Enter-key, c = 3 + Enter-key)
   - In same line (a, b, c = 1, 2, 3)

Operators and Operands

- *Operators* are functionality that do something and can be represented by the symbols such as + or special keywords
- Operators operate on data referred as operands
- For example:



Operators and Operands Exercises

Exercises: Assign numerical values to X and Y and do the following

1. Sum x and y by plus: x + y
2. Subtract y from x by minus: x - y
3. Multiply x and y: x * y
4. Multiply a string with y: 'X' * y
5. Return x to the power of y: x**y
6. Divide x by y: x/y
7. Return remainder of x divided by y: x % y
8. Check if x is greater than y: x > y
9. Check if x is smaller than y: x < y
10. Check if x is less than or equal to 100: x <= 100
11. Check if y is greater than or equal to 5: y >= 100
12. Check if x is equal to y: x == y
13. Check if x is not equal to y: x != y

Math Operators and Operands - 1

We can use multiple operators together

- Evaluation order is what high school taught us: BODMAS

- B: Brackets first
- O: Orders (i.e. Powers and Square Roots, etc.)
- DM: Division and Multiplication (left-to-right)
- AS: Addition and Subtraction (left-to-right)

- Division before multiplication and addition before subtraction
- However the order can be altered by using brackets
  - Example: 4/2*3+1-5 = 2 but 4/2*(3+1-5) = -2
  - Use multiple operators (example: x+y*y/x)

Math Operators and Operands - 2

- Operands can be over-written
  - Solution – 1
  - Assign a value to variable a
  - Reassign/overwrite value of a as a * 3

```
>>> a = 2
>>> a = a * 3
>>> a
6
```

Math Operators and Operands - 3

- Shortcuts for math operations:
  - Solution – 1
  - Assign a value to variable a
  - Reassign/overwrite value of a as a * 3

```
>>> a = 2
>>> a = a * 3
>>> a
6
```

  - Solution - 2

```
>>> a = 2
>>> a *= 3
>>> a
6
>>>
```

Data Structures

- *Data Structures* are containers that hold/store a collection of data/ object together
- There are two built-in data structures that we will discuss here

1. List (list): holds ordered collection of objects separated by comma
    - Objects are present in the given order any change in introduced
    - Lists are mutable: data can be added and removed
    - An empty list are created as: my_list = []
    - A list with values are created as: my_list = [1, 2, ‘a’, ‘b’]

Data Structures Exercises - 1

Exercises: print and check at each step

help(list)

1. create a list with five items and follow the exercise (hint: my_list = [1, 2, 'C', 4, 'E'])

2. Add/append an item: my_list.append('X')

3. Access the list item by index, which are the position of items (counted from 0)
   - Access the 1st item: my_list[0] (square brackets to define the index)
   - Access the last item: my_list[-1]
   - Access the 4th item (?)
   - Access items from position 2 to 4: my_list[1:4]
     - Here 4 means item in the 5th position, last mentioned index is not accessed
   - Access items at the index 2 to the second last position (?)
   - Access items from the beginning to the position 4: mylist[:4]
   - Access items at the index 2 to the last position (?)

4. Insert an item in the 4th position: my_list.insert(3, 'X')

Data Structures Exercises - 2

Exercises: print and check at each step

5. Check the items in the list and find the number of items: len(my_list)

6. Remove 'X' from the list: my_list.remove('X')
   - Check the length again

5. Remove an item from a position from any index (i=3): my_list.pop([i])

8. Get maximum value in the list: max(my_list)
9. Get minimum value in the list: min(my_list)
   - Letters are considered larger than digits

10. Reverse items in the list: my_list.reverse()
11. Sort items of the list: sorted(my_list)
    - Reverse the items again

12. Summed up the value of a list containing all the numerical items: sum(list_num)
    - Hint: list_num = [1, 2, 3, 4]

Data Structures Exercises - 3

Exercises: print and check at each step

13. Convert a string into list: list(my_string)
    - Hints: string = 'convert string into list'
14. Count the occurrence of an item
    - hint-1: list('convert string into list').count('t')
    - hint-2: new_list = list('convert string into list')
      new_list.count('t')
15. Convert a string into list by splitting it by space: my_string.split(' ')
    - Try splitting by different lists by different characters

Data Structures Exercises - 4

Exercises: print and check at each step

16. Get unique items of the list: set(new_list)
    - Note: Set is another data structute, with an unordered collection without duplicates Created as: my_set = set()

17. Dealing with two lists: define 2 lists with some items (list1 and list2)
    - Create a third list as list3: list3 = list1 + list2
    - Extend list1 by list2: list1.extend(list2)
    - Create a list with only unique items from the lists: set(list1).union(list2)
    - Find common items in the lists: set(list1).intersection(list2)

Data Structures

- *Data Structures* are containers that hold/store a collection of data/ object together
- There are two built-in data structures that we will discuss here

1. List (list): holds ordered collection of objects separated by comma
   - Objects are present in the given order any change in introduced
   - Lists are mutable: data can be added and removed
   - An empty list are created as: my_list = []
   - A list with values are created as: my_list = [1, 2, ‘a’, ‘b’]

2. Dictionary (dict): a list of key-value pairs where key can be any numbers or strings and values can be any arbitrary python object
   - An empty dict is created as: my_dict = {} or my_dict()
   - Key and value are separated by a colon (:)
   - A list with key-value pairs are created as: my_dict = {‘Key_1’: ‘Val_1’}

Data Structures Exercises - 5

Exercises: print and check at each step

help(dict)

1. Create a dictionary with key-value pairs: my_dict = {'name' : 'Khaleesi', 'age' : 20}
2. Access value by a key: my_dict['name'] or my_dict.get('name')
3. Add more items to the dictionary: my_dict['occupation'] = 'Queen'
4. Print all the items: my_dict
4. Print all the key-value pairs as list: my_dict.items()
5. Print all the keys of the dictionary as list: my_dict.keys()
6. Print all the values of the dictionary as list: my_dict.values()
7. Remove a key-value pair: my_dict.pop('age')
8. Remove the last key-value pair: my_dict.popitem()
9. Check if a key is in the dict: 'location' in my_dict and 'age' in my_dict
10. Remove all the items from the dict: my_dict.clear()

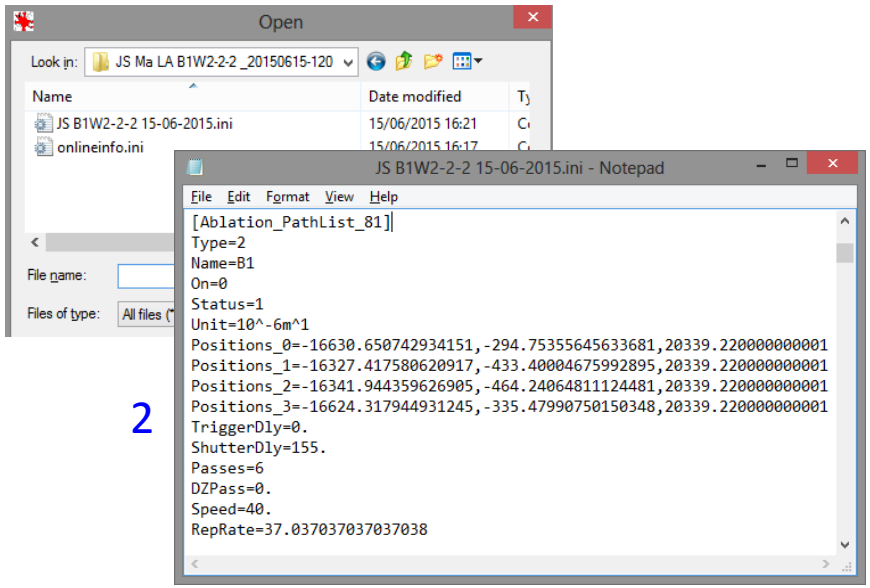
Task for the last session on Sept 24

- From your current scientific interests identify/create a task
  - That involves repeated tasks like reading one or multiple files of same format
  - Requires you to extract certain information
  - Requires processing of the data like using certain formula for calculations
  - Requires you to create a new file with the processed information

Task for the last session on Sept 24

- An example from Dr. Jan A. Schuessler:
  - He uses Laser Ablation System, that generates a text file containing several entries with x, y, z coordinates and other parameters
  - Task:
    1. Read the file contents
    2. Record information of each parameters for each entry
    3. Create an output file containing table with chosen information

1



2

3

Laser Ablation parameter
Pathlist Name: JS B1W2-2-2 16-06-2015
Overview Image: B1W2-2-2 2015-04-30

analysis no.	name	Trigger (s)	Shutter (s)	Δ_z/pass (µm)	Passes	Scan Speed (µm/s)	Raster Spacing (µm)	Dwell Time (s)	RepRate (Hz)
1	GOR132G	0	160	0	9	40	26	0	26
2	BHVO-2	0	160	0	9	40	26	0	84
3	GOR132G	0	160	0	9	40	26	0	26

Content of the slides by Dr. Jan A. Schuessler