

Intro. Comp. for Data Science (FMI08)

Dr. Nono Saha

April 12, 2023

Max Planck Institute for Mathematics in the sciences
University of Leipzig/SCADS.AI

Spring 2023

1. Introduction to Git
2. Essential commands for beginners
3. Introduction to Python
4. Basic types
5. Sequence types
6. Homework 2

Introduction to Git

What is Git?

- Modern version control system
- Mature, actively maintained, open source project (Linus Torvalds, 2005)
- Works well on a wide range of operating systems and IDEs

Main characteristics

- Distributed VCS vs CVC
- Performance: Algorithms, File content/names
- Security: top priority and SHA1 hashing algorithm for commits, content, file-folder
- Flexibility: small and large projects, many OS, branches and tags

Essential git commands

Init and clone directories

- **git init**: creates an empty Git repository
- **git clone**: clone a repository into a new directory
- More importantly learn to use the "manuel"

Basic commands

- **git pull** : get recent updates from the remote to local branch
- **git add** <file or folder>: add file contents to the index
- **git commit** -m <some comments>: record changes to the repository
- **git push** <origin> <branch>: update remote refs along with associated objects

More commands

- **git merge, git stash, git status, git log** <https://git-scm.com/docs>

Introduction to Python

What is Python?

- A programming language that boasts ease of use
- Dynamic typing and garbage-collected language
- Batteries included (pypi.python.org)

Several advantages

- Code readability with the use of significant indentation via the off-side rule
- High level and for general purposes
- Structured, functional and OO-programming

Important!!!!

- Network sockets, database handles, windows, file descriptors are not included in the garbage-collection
- Need of other methods (e.g. destructors).

Variables: basic types

What is a variable?

- A value stored in computer memory.
- It should have a name and the value stored.
- Use a combination of alphanumeric characters and the underscore character for names
- Convention recommends lowercase characters with words separated by underscore for readability

Type system

- **Boolean** (or **bool**): e.g. True, False
- **Integer** (or **int**): e.g. 1, 34
- **Float**: e.g. 1., 1.0, 2.4
- **Complex Number** (or **complex**): e.g. $1+1j$, $1+0j$, $4j$, etc...

Variable: dynamic types

So most basic operations will attempt to force variable to a consistent type appropriate for the operation.

Boolean operations and comparisons

```
1      1 and True          5. > 1
2      ## True             ## True
3      0 or 1              5. == 5
4      ## 1                ## True
5      not 0               1 > True
6      ## True             ## False
7      not (0+0j)          (1+0j) == 1
8      ## True             ## True
9      not (0+1j)          'abc' < "ABC"
10     ## False            ## False
11
```

Now, what if I do?

```
1      "abc" > 5
2
```

Variables: basic mathematical operations

Boolean operations and comparisons

```
1      1 + 5          5 / 1.  
2      ## 6          ## 5.0  
3      1 + 5.  
4      ## 1          ## 2.5  
5      1 * 5.  
6      ## 5.0        ## 2  
7      True * 5      5 % 2  
8      ## 5          ## 1  
9      (1 + 0j) - (1 + 1j) 7 ** 2  
10     ## -1j         ## 49  
11
```

Now, what if I do?

```
1      "abc" + 5      ## Error? what type of error? why?  
2      "abc" + str(5) ## does that correct the prev error?  
3      "abc" ** 2     # what about this?  
4      "abc" * 3      # and this?  
5
```


Variables: casting and assignment

Casting using type functions: e.g. float(), int(), etc..

```
1 float ("0.5")          bool(0)
2 ## 0.5                 ## False
3 float(True)            bool("hello")
4 ## 1.0                 ## ??
5 int(1.1)               str(3.14159)
6 ## 1                   ## "3.14159"
7 int("2")              str(True)
8 ## 2                   ## "True"
```

Now, what if I do?

```
1 int("2.1")
2 ## Error? what type of error? why?
```

Variable assignment

```
1 x = 100 ## Assign a value of 100 to a variable named x
2 a = b = 5 ## Assign a value of 5 to both variable a and b
```

Variables: special values

No missing values and non-finite floating point values are available. There is a None type similar to NULL in R, Java, JavaScript.

```
1  1 / 0
2  ## Error in py_call_impl(callable, dots$args, dots$keywords)
3  : ZeroDivisionError: division by zero
4  ## Detailed traceback:
5  ##   File "<string>", line 1, in <module>
6
7  1. / 0
8  ## Error in py_call_impl(callable, dots$args, dots$keywords)
9  : ZeroDivisionError: float division by zero
10 ## Detailed traceback:
11 ##   File "<string>", line 1, in <module>
12
13 float("nan")
14 ## nan
15 float("-inf")
16 ## -inf, we can do 5 > float("inf")
```

Variables: string literals

Strings can be defined using a couple of different ways,

```
1 'allows embedded "double" quotes'  
2 ## 'allows embedded "double" quotes'  
3
```

```
1 "allows embedded 'single' quotes"  
2 ## "allows embedded 'single' quotes"  
3
```

strings can also be triple quoted, using single or double quotes, which allows the string to span multiple lines.

```
1 """line one  
2 line two  
3 line three"""  
4 ## 'line one\nline two\nline three'  
5
```

Several methods possible:

```
1 x = "Hello wolrd! 1234"  
2 x.find("!"), x.isalnum(), x.title(), x.swapcase(), x.split()
```

Variables: sequence types

lists in Python

Python lists are a heterogenous, ordered, mutable containers of objects (they behave very similarly to lists in R).

```
1  x = [0,1,1,0], x
2  ## [0, 1, 1, 0], we can subsetting with x[start:stop:step]
3
4  [0, True, "abc"]
5  ## [0, True, 'abc'] mutate an element, x[-1] = 2
6
7  [0, [1,2], [3,[4]]]
8  ## [0, [1, 2], [3, [4]]], can we assign? copy and deepcopy
9
10 x = [0,1,1,0]
11 type(x)
12 ## <class 'list'> we can sort with x.sort()
13
14 y = [0, True, "abc"]
15 type(y)
16 ## <class 'list'> is x.sort() still work?
17
```

Variables: sequence types

Unpacking lists in Python

Unpacking into multiple variables when doing assignment,

```
1      x, y = [1,2]
2      x
3      ## 1 similarly we can do x, y = [[0,1], [2, 3]]
4      y
5      ## 2
6      x, y = [1, [2, 3]]
7      x
8      ## 1 or something like (x1,y1), (x2,y2) = [[0,1], [2, 3]]
9      y
10     ## [2, 3]
11
```

Extended unpacking:

```
1      x, *y = [1,2,3] ## what about this x, y = [1,2,3]?
2      y ## [2, 3] what about x*, y = [1,2,3]?
3
```

Variables: sequence types

Tuples in Python

Python tuples are a heterogenous, ordered, immutable containers of values.

They are nearly identical to lists except that their values cannot be changed.

```
1      (1,2,3)
2      ## (1, 2, 3)
3
4      (1,True,"abc")
5      ## (1, True, 'abc')
6
7      (1,(2,3))
8      ## (1, (2, 3))
9
10     x = (1,2,3)
11     x[2] = 5 ## what will happen here? and what about this
del x[2]?
```

Variable: ranges as a sequence type

These are the last type sequence type and are a bit special - ranges are a homogenous, ordered, immutable "containers" of integers.

Examples:

```
1  range(10)
2  ## range(0, 10)
3
4  range(0,10)
5  ## range(0, 10)
6
7  range(0,10,2)
8  ## range(0, 10, 2)
9
10 range(10,0,-1)
11 ## range(10, 0, -1)
12
13 list(range(10))
14 ## [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] what about this list(range
15 (10,0,-1))
```

What next?

Set and Mapping types

We will discuss sets (`set`) and dictionaries (`dict`) in more detail next week.

Specifically we will discuss the underlying data structure behind these types (as well as `lists` and `tuples`) and when it is most appropriate to use each.

Homework1: programming like a hispter!

- Write a program that compute a square root of an interger. NB: making use of no python library.
- Given a list, write a program that returns an ascendent sorted list.