Intro. Comp. for Data Science (FMI08)

Dr. Nono Saha

April 13, 2023

Max Planck Institute for Mathematics in the Sciences
University of Leipzig/ScaDS.AI

Spring 2023

Course plan

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 ”manual.”

Basic commands

- **git pull**: get recent updates from the remote to the 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 fetch**, **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, and 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 force a variable to a consistent type appropriate for the operation.

Boolean operations and comparisons

```
1 and True                  5. > 1
## True                     ## True
0 or 1                      5. ==  5
## 1                        ## True
not 0                       1 > True
## True                     ## False
not (0+0j)                  (1+0j) == 1
## True                     ## True
not (0+1j)                 'abc' < "ABC"
## False                    ## False

```

Now, what if I do?

```
"abc" > 5

```

Variables: basic mathematical operations

Math operations

```
1 + 5                    5 / 1.
## 6                     ## 5.0
1 + 5.                   5 / 2
## 6.                      ## 2.5
1 * 5.                   5 // 2
## 5.0                   ## 2
True * 5                 5 % 2
## 5                     ## 1
(1 + 0j) - (1 + 1j)      7 ** 2
## -1j                   ## 49

```

Now, what if I do?

```
"abc" + 5   ## Error? What type of error? Why?
"abc" + str(5) ## Does that correct the prev error?
"abc" ** 2 # What about this?
"abc" * 3  # And this?

```

Variables: casting and assignment

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

```
float ("0.5")
## 0.5
float(True)
## 1.0
int(1.1)
## 1
int("2")
## 2

```

```
bool(0)
## False
bool("hello")
## ??
str(3.14159)
## "3.14159"
str(True)
## "True"

```

Now, what if I do?

```
int("2.1")
## Error? What type of error? Why?

```

Variable assignment

```
x = 100 ## Assign a value of 100 to a variable named x
a = b = 5 ## Assign a value of 5 to both variables 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 / 0
## Error in py_call_impl(callable, dots$args, dots$keywords)
: ZeroDivisionError: division by zero
## Detailed traceback:
##   File "<string>", line 1, in <module>

1. / 0

## Error in py_call_impl(callable, dots$args, dots$keywords)
: ZeroDivisionError: float division by zero
## Detailed traceback:
##   File "<string>", line 1, in <module>

float("nan")
## nan
float("-inf")
## -inf, we can do 5 > float("inf")

```

Variables: string literals

Strings can be defined using a couple of different ways,

```
'allows embedded "double" quotes'
## 'allows embedded "double" quotes'

```

```
"allows embedded 'single' quotes"
## "allows embedded 'single' quotes"

```

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

```
"""line one
line two
line three"""
## 'line one\nline two\nline three'

```

Several methods are possible:

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

```

Variables: sequence types

Lists in Python

Python lists are heterogenous, ordered, mutable containers of objects.

```
    x = [0,1,1,0]
    x
    ## [0, 1, 1, 0], we can use subsetting with x[start:stop:
  step]
    [0, True, "abc"]
    ## [0, True, 'abc'] mutate an element, x[-1] = 2
    [0, [1,2], [3,[4]]]
    ## [0, [1, 2], [3, [4]]], can we assign? Copy and deepcopy
    x = [0,1,1,0]
    type(x)
    ## <class 'list'> we can sort with x.sort()
    y = [0, True, "abc"]
    type(y)
    ## <class 'list'> is y.sort() still work?

```

Variables: sequence types

Unpacking lists in Python

Unpacking into multiple variables when doing ”assignment”,

```
x, y = [1,2]
x
## 1 similarly we can do x, y = [[0,1], [2, 3]]
y
## 2
x, y = [1, [2, 3]]
x
## 1 or something like (x1,y1), (x2,y2) = [[0,1], [2, 3]]
y
## [2, 3]

```

Extended unpacking:

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

```

Variables: sequence types

Tuples in Python

Python tuples are heterogenous, ordered, immutable (or non-mutable) containers of values.

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

```
(1,2,3)
## (1, 2, 3)

(1,True,"abc")
## (1, True, 'abc')

(1,(2,3))
## (1, (2, 3))

x = (1,2,3)
x[2] = 5 ## What will happen here? And what about this del x[2]?

```

Variable: ranges as a sequence type

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

Examples:

```
range(10)
## range(0, 10)

range(0,10)
## range(0, 10)

range(0,10,2)
## range(0, 10, 2)

range(10,0,-1)
## range(10, 0, -1)

list(range(10))
## [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]. What about this list(range(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 hipster!

- Write a program that computes a square root of any given integer. NB: making use of no Python library.
- Given a list, write a program that returns an ascendent sorted list.