LABORATORY OF DATA SCIENCE

Python recap

Python

Python is a

- ¤ High-level
- ¤ Interpreted (Interpreters for many OS)
- ¤ Dynamically Typed
    - ■ Verification of the type safety of a program at runtime
- ¤ Object oriented
- ¤ Cross-Platform
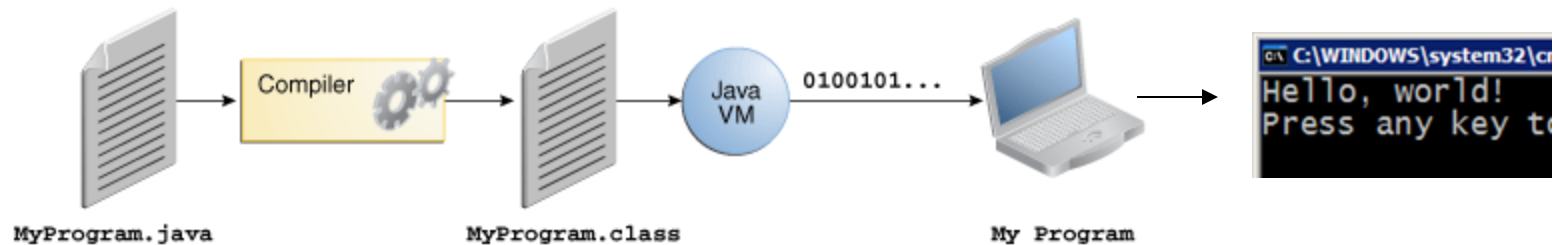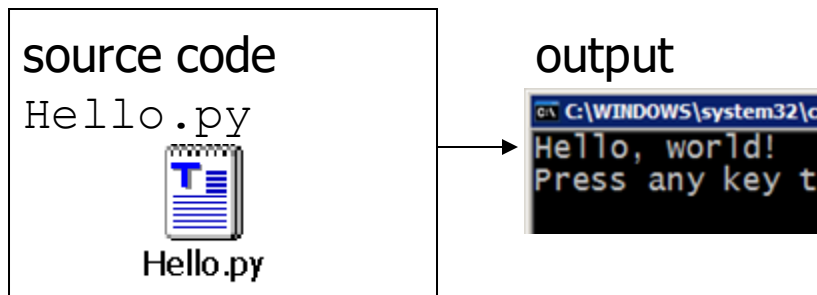- ¤ Multi-purpose (WEB, GUI, Scripting)

computer programming language

https://www.python.org/

Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.

MyProgram.java → Compiler → MyProgram.class → Java VM → 0100101... → My Program → C:\WINDOWS\system32\cmd Hello, world! Press any key to

interpret

source code
Hello.py

Hello.py

output

C:\WINDOWS\system32\cmd
Hello, world!
Press any key to

- Python is instead directly *interpreted* into machine instructions.

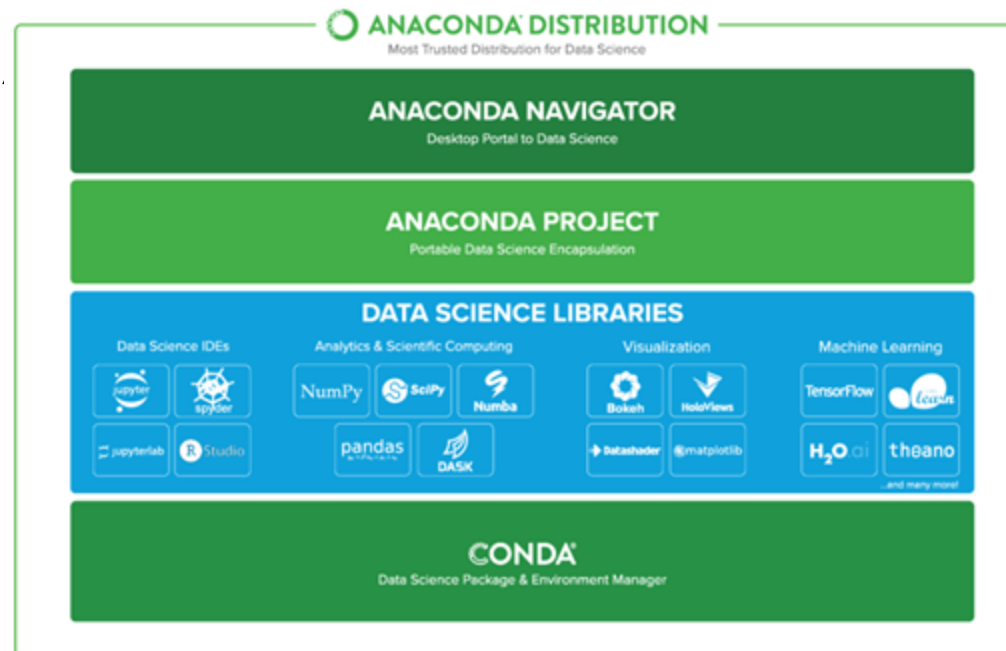# Python language: books
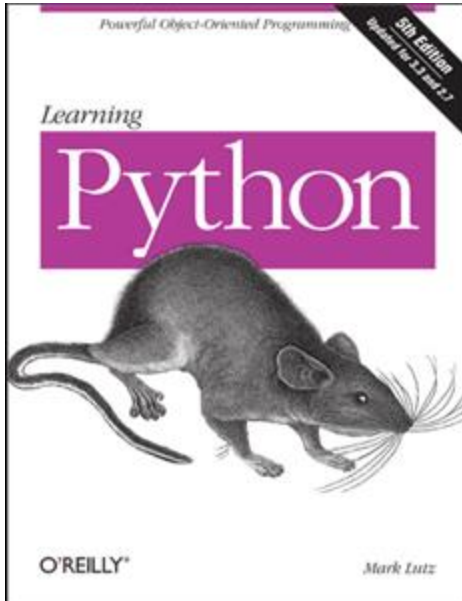
http://www.spronck.net/pythonbook/

Anaconda - www.anaconda.com

- Manage your DS packages, **dependencies**, and environments

- Develop DS projects using Jupyter, JupyterLab, Spyder…

- Automatically manages all packages, including cross-language **dependencies**

- Works across all platforms: Linux, macOS, Windows

Anaconda Navigator

- Desktop Portal to Data Science

- Install and launch applications and editors including Jupyter, RStudio, Visual Studio Code, Spyder…

- Manage your local environments and data science projects from a graphical interface

8

Python Recap

Lab of Data Science
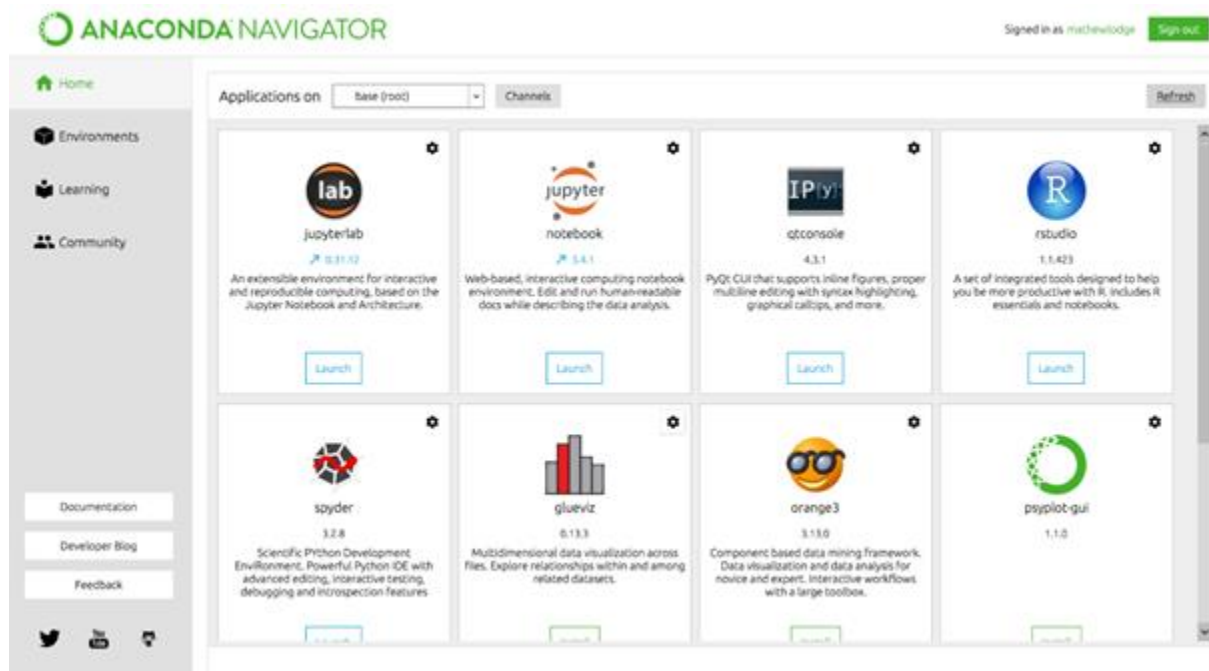
Indentation

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

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

Numbers

```python
# 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")
```

Lab of Data Science

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
```

Lab of Data Science

Strings

```
#This is a string
name = 'Anna Monreale (that\'s not me)'

#This is also a string
city = "Pisa"

#This is a multi-line string
office = """My office is at the department
of Computer Science, University of Pisa"""

#This is also  a multi-line string
other = '''My office hours is on Tuesday in the
afternoon, however, it is always better to take
an appointment'''
```

String manipulation

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

fruits = ', '.join(['Apples', 'Bananas', 'Oranges'])
# Apples, Bananas, Oranges

end_of_the_world = "%s %d %d" % ('Dec', 21, 2012)
# Dec 21 2012

#This is also  a multi-line string
other = f"On {end_of_the_world} I ate {fruits}"
# On Dec 21 2012 I ate some apples, bananas, oranges
```

Lists

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

# Appending
favorites.append(42)

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

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

Lab of Data Science

Lists

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

len(numbers)
# 5

numbers[0]
# 1

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

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

Lab of Data Science

Dictionary

```python
person = {}

# Set by key / Get by key
person['name'] = 'Nowell Strite'

# Update
person.update({
    'favorites': [42, 'food'],
    'gender': 'male',
    })

# Any immutable object can be a dictionary key
person[42] = 'favorite number'
person[(44.47, -73.21)] = 'coordinates'
```

Lab of Data Science

Dictionary

```python
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']
person.get('name', 'Anonymous')
# 'Nowell Strite'

person.keys()
# ['name', 'gender']

person.values()
# ['Nowell', 'Male']

person.items()
# [['name', 'Nowell'], ['gender', 'Male']]
```

Set

```python
set_a = {1, 2, 3, 4}
set_b = {3, 4, 5}

set_a.union(set_b) # set_a | set_b
# {1, 2, 3, 4, 5}

set_a.intersection(set_b) # set_a & set_b
# {2, 4}

set_a.difference(set_b) # set_a - set_b
# {1, 2}

set_a.pop()
# 1
set_a
# {2, 3, 4}
```

Lab of Data Science

Additional built-in Functions

```
a = {4, 3, 2, 1, 0}

sorted(a)
# [0, 1, 2, 3, 4]

min(a)  # max
# 0

len(a)
# 5

sum(a)
# 10


# And more...
```

Lab of Data Science

If-then-else

```
grade = 82
if grade >= 90:
        if grade == 100:
                print 'A+'
        else:
                print "A"
elif grade >= 80:
        print "B"
elif grade >= 70:
        print "C"
else:
        print "F"

# B
```

Lab of Data Science

For Loop

```python
for x in range(10):  #0-9
    print(x)
```

```python
fruits = ['Apple', 'Orange']

for fruit in fruits:
    print(fruit)
```

```python
states = {
    'VT': 'Vermont',
    'ME': 'Maine',
    }

for key, value in states.items():
    print('%s: %s' % (key, value))
```

Function Definition

```python
def my_function():
    """Function Documentation"""
    print("Hello World")
```

```python
# Positional
def add(x, y):
    return x + y

# Keyword
def shout(phrase='Yipee!'):
    print(phrase)

# Positional + Keyword
def echo(text, prefix=''):
    print('%s%s' % (prefix, text))
```

Lab of Data Science

Exercise: maximal subsequence

Given an array of integers, e.g.

¤ a = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

And a function S to compute the sum from h to k,

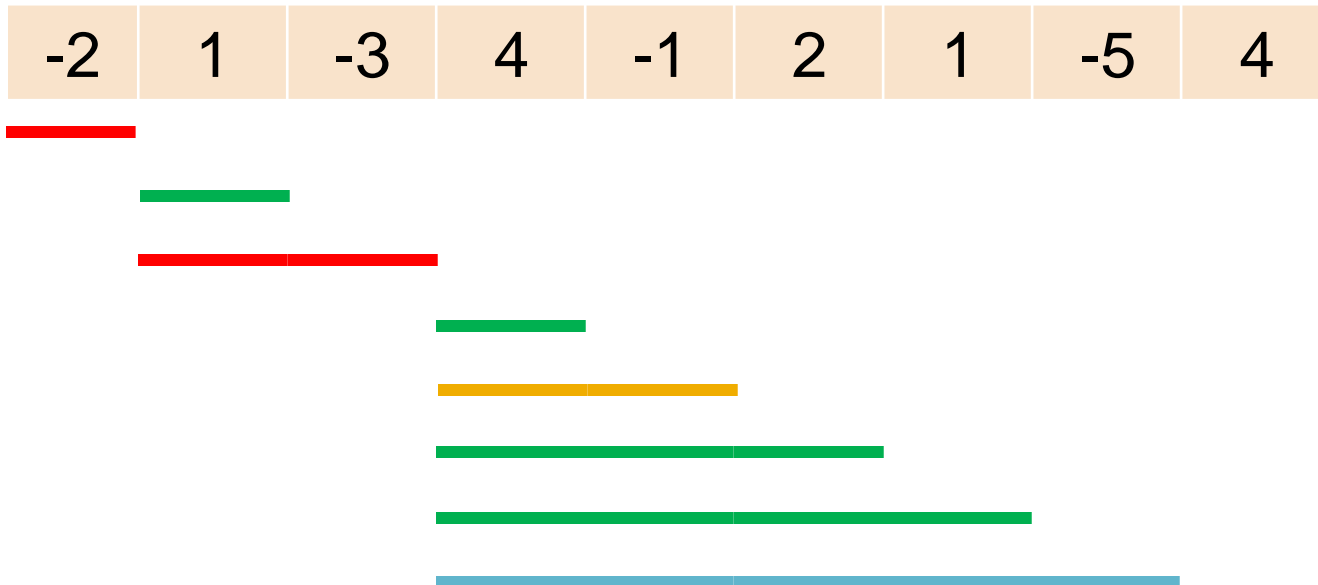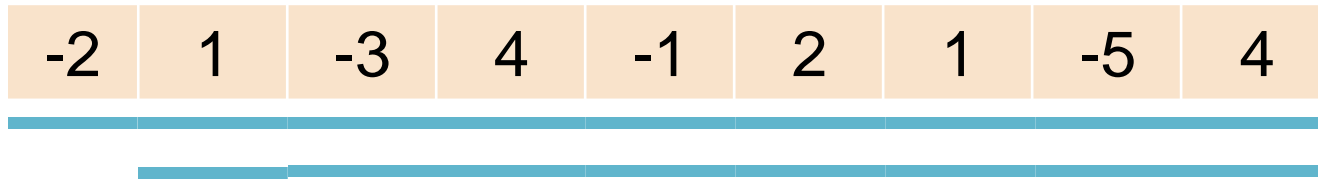$$S(a, h, k) = \sum_{i=h}^{k} a[i]$$

Find the values of h and k such that S maximizes ($S(3, 6) = 6$)

Variant:

- Use `a = generate_large_array()` (import the function from the provided `supplementary_code.py`) and make the code run in less then 1s

Exercise: maximal subsequence

-2	1	-3	4	-1	2	1	-5	4

-2	1	-3	4	-1	2	1	-5	4

current	max
0	0
1	1
0	1
4	4
3	4
5	5
6	6
1	6

Lab of Data Science

For more info google **Segment Tree**

Exercise: lists and dictionaries

- Given the list: l = [12, 3, -4, 6, -5, 9]
- Given the dictionary:
  - d = {'apple': 3, 'orange': 4, 'tomato': -5, 'meat': 6, 'potato': 15, 'strawberry': 9}
- If a value in the dictionary is found in the list, add the corresponding key to a string named 'to_buy' and print it at the end.
- If a value in the dictionary is not found in the list, chose a random value from the list, that is not present in the dictionary, and assign it to the corresponding key. Print the updated dictionary at the end.

Exercise: lists

- Given 2 lists:

a = [12, 3, 4, 6, 5, 9]

b = [10, 3, 2, 6, 3, 7]

Compute the Pearson's correlation.

Exercise: lists (hint)

□ Given 2 lists:

a = [12, 3, 4, 6, 5, 9]

b = [10, 3, 2, 6, 3, 7]

Compute the Pearson's correlation.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Where σ is the *std deviation*

Exercise: for loops

- Import and run `generate_order` and `get_menu` from `sushi_rest.py`
  - `Generate_order` return the list of all the plates in a order
  - `Get_menu` return a dictionaty with `<plate_name, price>`

- Answer the following questions:
  - How many plates are in the order?
  - How many unique plates are in the order?
  - For each plate in the menu, find if it is in the order

  - Create a dictionary that counts how often each plate from the menu appears in the order.
  - Add to the previous dictionary all menu plates, even if they don't appear in the order.

Exercise: for loops (with constraints)

- Import and run `generate_order` and `get_menu` from `sushi_rest.py`
  - `Generate_order` return the list of all the plates in a order
  - `Get_menu` return a dictionaty with `<plate_name, price>`

- Answer the following questions:
  - How many plates are in the order? **Max 1 instruction**
  - How many unique plates are in the order? **Max 2 instruction**
  - For each plate in the menu, find if it is in the order
    Less then len(menu)*len(order) iterations
  - Create a dictionary that counts how often each plate from the menu appears in the **len(order) iterations**
  - Add to the previous dictionary all menu plates, even if they don't appear in the order. **Use the for loop to add elements to the dictionary ONLY. Max 1 for loop**
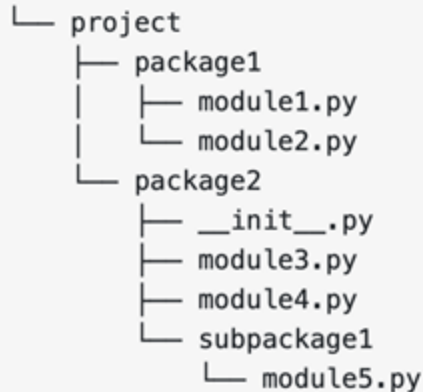
Import packages

```python
# Renaming imports
from datetime import date
from my_module import date as my_date

# This is usually considered a big No-No
from datetime import *
```

Kinds of Imports

```
└── project
    ├── package1
    │   ├── module1.py
    │   └── module2.py
    └── package2
        ├── __init__.py
        ├── module3.py
        ├── module4.py
        └── subpackage1
            └── module5.py
```

```python
from package1 import module1
from package1.module2 import function1
from package2 import class1
from package2.subpackage1.module5 import function2
```

Absolute

```python
# package1/module1.py
from .module2 import function1
```

```python
# package2/module3.py
from . import class1
from .subpackage1.module5 import function2
```

Relative

Error Handling

```python
import datetime
import random

day = random.choice(['Eleventh', 11])
try:
    date = 'September ' + day
except TypeError:
    date = datetime.date(2010, 9, day)
else:
    date += ' 2010'
finally:
    print(date)
```
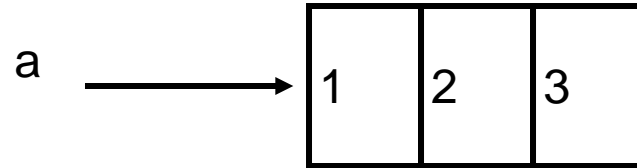
Lab of Data Science

Reference Semantics

- Assignment manipulates references
  - x = y **does not make a copy** of y
  - x = y makes x **reference** the object y references
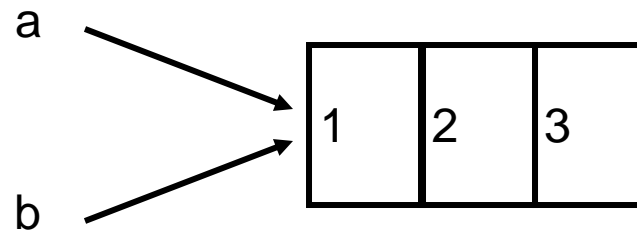- Very useful; but beware!
- Example:
  ```
  >>> a = [1, 2, 3]
  >>> b = a
  >>> a.append(4)
  >>> print b
  [1, 2, 3, 4]
  ```
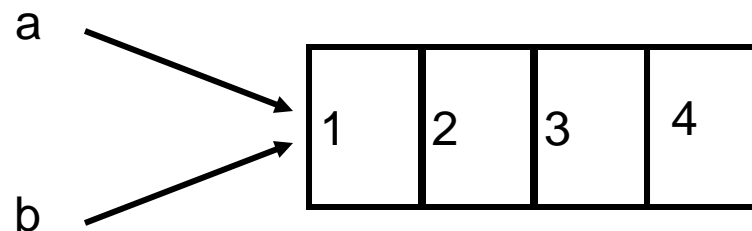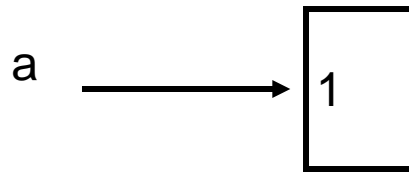
Changing a Shared List

a = [1, 2, 3]

a ⟶ | 1 | 2 | 3 |
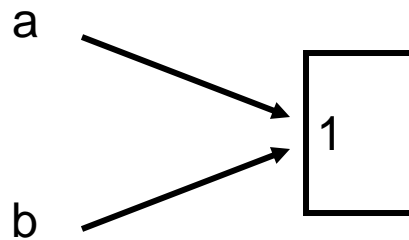
b = a

a ↘
| 1 | 2 | 3 |
b ↗

a.append(4)

a ↘
| 1 | 2 | 3 | 4 |
b ↗

Changing an Integer

a = 1

a ───────────▶ | 1 |

b = a

a ╲
   ╲
    ▶ | 1 |
   ╱
b ╱

new int object created
by add operator (1+1)

a = a+1

a ───────────▶ | 2 |
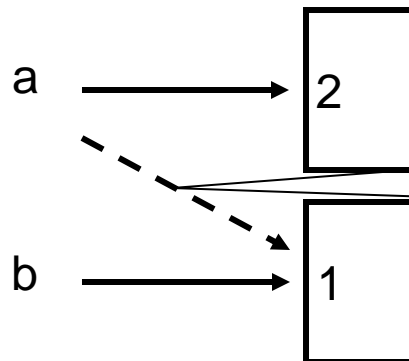
old reference deleted
by assignment (a=...)

b ───────────▶ | 1 |