Python Workshop

Jingsai Liang & Greg Gagne

Jan 14, 2021

Variable

Declaration of variables is not required in Python.

- integer
  - The size of interger could be unlimited. No need to worry about integer overflow.

```
>>> i = 2**100
>>> i
1267650600228229401496703205376
```

  - Support multiple bases

```
>>> i, j, k = 0b100, 0o11, 0xAA
>>> print(i,j,k)
4 9 170
```

Variable

- float
- boolean
  - True, False
- complex number

```
>>> import math, cmath
>>> x = math.sqrt(-1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
>>> x = cmath.sqrt(-1)
>>> type(x)
<class 'complex'>
>>> x = 2 + 3j
>>> x
(2+3j)
```

Operation

- Addition/Substration/Multiplication - same as Java/C
- Division
  - python 2: 1/2 = 0
  - python 3: 1/2 = 0.5
- Floor/Integer Division
  - python 3: 1//2 = 0, -23//10 = -3 (round to floor)
  - Java/C: -23/10 = -2 (round to zero)
- Modulus
  - python 3: -23%10 = 7 (-23 = -3 * 10 + 7)
  - Java/C: -23%10 = -3 (-23 = -2 * 10 - 3)

Assignment

Practice 1:

Write a piece of pseudocode to swap values of two variables a and b.

Assignment

Code Practice 1:

Write a piece of pseudocode to swap values of two variables a and b.

Java/C solution:

```
tmp = a
a = b
b = tmp
```

Assignment

Code Practice 1:

Write a piece of pseudocode to swap values of two variables a and b.

Java/C solution:

```
tmp = a
a = b
b = tmp
```

Python solution:

```
a,b = b,a
```

Assignment

Question 1:

Given a singly linked list, insert a node *p* after current node *cur*.

Java/C solution:

```
tmp = cur.next
cur.next = p
p.next = tmp
```

In python, we only need a one-line code to insert *p* after *cur*.

Assignment

Question 1:

Given a singly linked list, insert a node *p* after current node *cur*.

Java/C solution:

```
tmp = cur.next
cur.next = p
p.next = tmp
```

In python, we only need a one-line code to insert *p* after *cur*.

```
cur.next, p.next = p, cur.next
```

List

- Initialization/Generation

```
>>> [1,2,3] # one dimension
[1, 2, 3]
>>> [[1,2], [3,4]] # two dimensions, list of list
[[1, 2], [3, 4]]
>>> [1.0, 2, 'str'] # each element can be any type of object
[1.0, 2, 'str']
```

range(start, end, stride)

```
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range(1,5))
[1, 2, 3, 4]
>>> list(range(1,10,2))
[1, 3, 5, 7, 9]
```

List

- Initialization/Generation

```
>>> [1, 2, 3] + [4, 5] # concatenation
[1, 2, 3, 4, 5]

>>> [2] * 3 # duplication
[2, 2, 2]

>>> [1,2,3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

List

- Slicing
  - index:
    - [0, 1, 2, …. , len(list)-2, len(list)-1]
    - [0, 1, 2, …. , -2, -1]

```
>>> x = list(range(10))
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> x[0], x[2], x[-1], x[-3]
(0, 2, 9, 7)
```

List

- Slicing
  - [start : end: stride]
    - default stride is 1
    - if stride > 0:
      - for(i = start; i < end; i += stride)
      - default start is the first one
      - default end is the last one

List

- Slicing if stride > 0

```
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> x[1:4] # 1:4:1, defaut stride is 1
[1, 2, 3]

>>> x[:4] # 0:4:1, default start is the first one
[0, 1, 2, 3]

>>> x[4:] # 4:11:1, default end is the last one
[4, 5, 6, 7, 8, 9]
```

List

- Slicing
  - [start : end: stride]
    - if stride < 0:
      - for(i = start; i > end; i += stride)
      - default start is the last one
      - default end is the first one

List

- Slicing if stride < 0

```
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> x[6:1:-1]
[6, 5, 4, 3, 2]
>>> x[6::-1] # default end is the first one
[6, 5, 4, 3, 2, 1, 0]
>>> x[:6:-1]# 9:6:-1 or -1:-4:-1, default start is the last one
[9, 8, 7]
```

Code Practice 2:
Can we reverse a list through slicing? How?

List

- Slicing

  Code Practice 2:
  Can we reverse a list through slicing? How?

```
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> x[::-1] # -1:-11:-1
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

List

- Method
  - len: `len(x)`
  - append: `x.append(3)`
  - pop: `x.pop(), x.pop(0)`
  - insert: `x.insert(0,7)`
  - count: `x.count(3)`
  - index: `x.index(3)`
  - sort `x.sort()`

Question 2:
Can we use a list as a stack or a queue? How?

List

- Method

  Question 2:
  Can we use a list as a stack or a queue? How?

  - stack: LIFO
    - push: append(e)
    - pop: pop()
  - queue: FIFO
    - add/remove: append(e) / pop(0)
    - add/remove: insert(0, e) / pop()

List

- mutable

```
>>> x = [1,2,3]
>>> x[0] = 0
>>> x
[0, 2, 3]
```

pass by reference

```
>>> def change(x): x[0]=0

>>> x=[1,2,3]
>>> change(x)
>>> x
[0, 2, 3]
```

List

- copy

  Dangerous Zone

```
>>> x = [1,2,3]
>>> y = x
>>> x[0] = 0
>>> x
[0, 2, 3]
>>> y
????
```

List

- copy

  Dangerous Zone

```
>>> x = [1,2,3]
>>> y = x
>>> x[0] = 0
>>> x
[0, 2, 3]
>>> y
[0, 2, 3] # copy by reference
```

  - copy by value

```
>>> x = [1,2,3]
>>> y = x[:]
>>> y = x.copy()
>>> y = list(x)
```

List

- copy

  Dangerous Zone

```
>>> x
[[1, 2], [3, 4]]
>>> y = x[:]
>>> x[0][0] = 0
>>> x
[[0, 2], [3, 4]]
>>> y
???
```

List

- copy

  Dangerous Zone

```
>>> x
[[1, 2], [3, 4]]
>>> y = x[:]
>>> x[0][0] = 0
>>> x
[[0, 2], [3, 4]]
>>> y
[[0, 2], [3, 4]] # shallow copy
```

  - deep copy

```
>>> from copy import deepcopy as copy
>>> y = copy(x)
```

String

- immutable

```
>>> s = 'hello'
>>> s[0] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

- slicing: same as list

```
>>> s[::-1]
'olleh'
```

- concatenation

```
>>> 'Westminster' + 'College' + str(100)
'WestminsterCollege100'
```

String

- method
  - str.strip(), str.split(pattern), pattern.join(str)

```
>>> path = '  doc/cmpt/360/final  '

>>> path = path.strip()
>>> path
'doc/cmpt/360/final'

>>> path = path.split('/')
>>> path
['doc', 'cmpt', '360', 'final']

>>> path = '/'.join(path)
>>> path
'doc/cmpt/360/final'
```

tuple

- immutable
  tuple is similar with list, but it is immutable.

```
>>> t = (1,2,3)
>>> t[0] = 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

- create a tupe with only one element

```
  >>> (1) # Wrong.
  1
  >>> (1,) # correct
  (1,)
  >>> 1, # correct
  (1,)
```

tuple

- pack and unpack

```
>>> t = 3,4 # packing
>>> t
(3, 4)
>>> a,b = t # unpacking
>>> a
3
>>> b
4
```

- concatenation

```
>>> t + (5,6)
(3, 4, 5, 6)

>>> t + tuple([7,8])
(3, 4, 7, 8)
```

dictionary

- create

  dictionary (hash tables) is mutable.

```
>>> d = {} # empty dictionary
>>> d['x'] = 3 # add one item
>>> d['y'] = 5
>>> d['x'] = 4 # change the value of key x
>>> d
{'x': 4, 'y': 5}

>>> d2 = {'x':7, 'y':8} # key:value
>>> d2
{'x': 7, 'y': 8}
```

dictionary

- key, value

```
>>> len(d) # number of keys
2

>>> list(d.keys())
['x', 'y']
>>> list(d.values())
[7, 8]
>>> list(d.items())
[('x', 7), ('y', 8)]

>>> 'y' in d # check if key y in d
True

>>> for key in d: print(key, d[key]) # loop over d
x 7
y 8
```

set

- create

  No duplication. No order.

```
>>> s={1,2,3,4,4,4,4}
>>> s
{1, 2, 3, 4}

>>> s = set([1,2,3,4,4,4]) # create a set from a list
>>> s
{1, 2, 3, 4}
```

Code Practice 3
Given a list, can we use set to remove all duplications from this list?

set

- create

  Code Practice 3
  Given a list, can we use set to remove all duplications from this list?

```
>>> a = [1,2,2,3,4,2,1,5]
>>> list(set(a))
[1, 2, 3, 4, 5]
```

set

- operation

```
>>> s1 = {1,2,3,4}
>>> s2= {3,4,5}

>>> s1 & s2 # intersection, and
{3, 4}
>>> s1 | s2 # union, or
{1, 2, 3, 4, 5}
>>> s1 ^ s2 # XOR, exclusive or
{1, 2, 5}
>>> s1 - s2 # difference
{1, 2}
>>> s2 - s1 # difference
{5}
```

function

- Python function can return multiple values as a tuple.

```
def f(a = 0): # default value of a is 0
    return a, a+1, a+2

>>> a, b, c = f(3)
```

- pass statement: placeholder for future implementation.

```
def future():
    pass
```

function

- scope:
  - global

```
a = 0
def my_function():
    print(a) # print global a

my_function()
```

  - local

```
a = 0
def my_function():
    a = 3 # create a local a which always takes precedence
    print(a) # print local a

my_function()
print(a) # print global a
```

function

- scope

Code Practice 4

Run this piece of code and report what is the ouput:

````
Is this correct?

```python
a = 0

def my_function():
    print(a)
    a = 3
    print(a)

my_function()
```
````

function

- scope
  Code Practice 4
  Is this correct?

```
a = 0

def my_function():
    # Should refer to local a, but a has not been defined.
    print(a)
    # Create a local a.
    # We cannot refer global a elsewhere in this function.
    a = 3
    print(a)

my_function()

>>> UnboundLocalError: local variable 'a' referenced
    before assignment
```

for loop

Java/C++ style:

```
for(int i = 0; i < 100: i ++){
    cout<< list[i] << endl;
}
```

python style:

```
for index in range(len(list):
    print(list[index])
```

or

```
for element in list:
    print(element)
```

if condition

```
if cond1 and cond2:
    statement
elif not cond3:
    statement
else:
    statement
```

Check if a list is empty

```
if len(list) != 0:
    do something
```

A better way:

```
if list:
   do something
```

if condition

Check if x is in path

```
for i in path:
    if i == 'x':
        do something
```

A better way:

```
if 'x' in path:
   do something
```

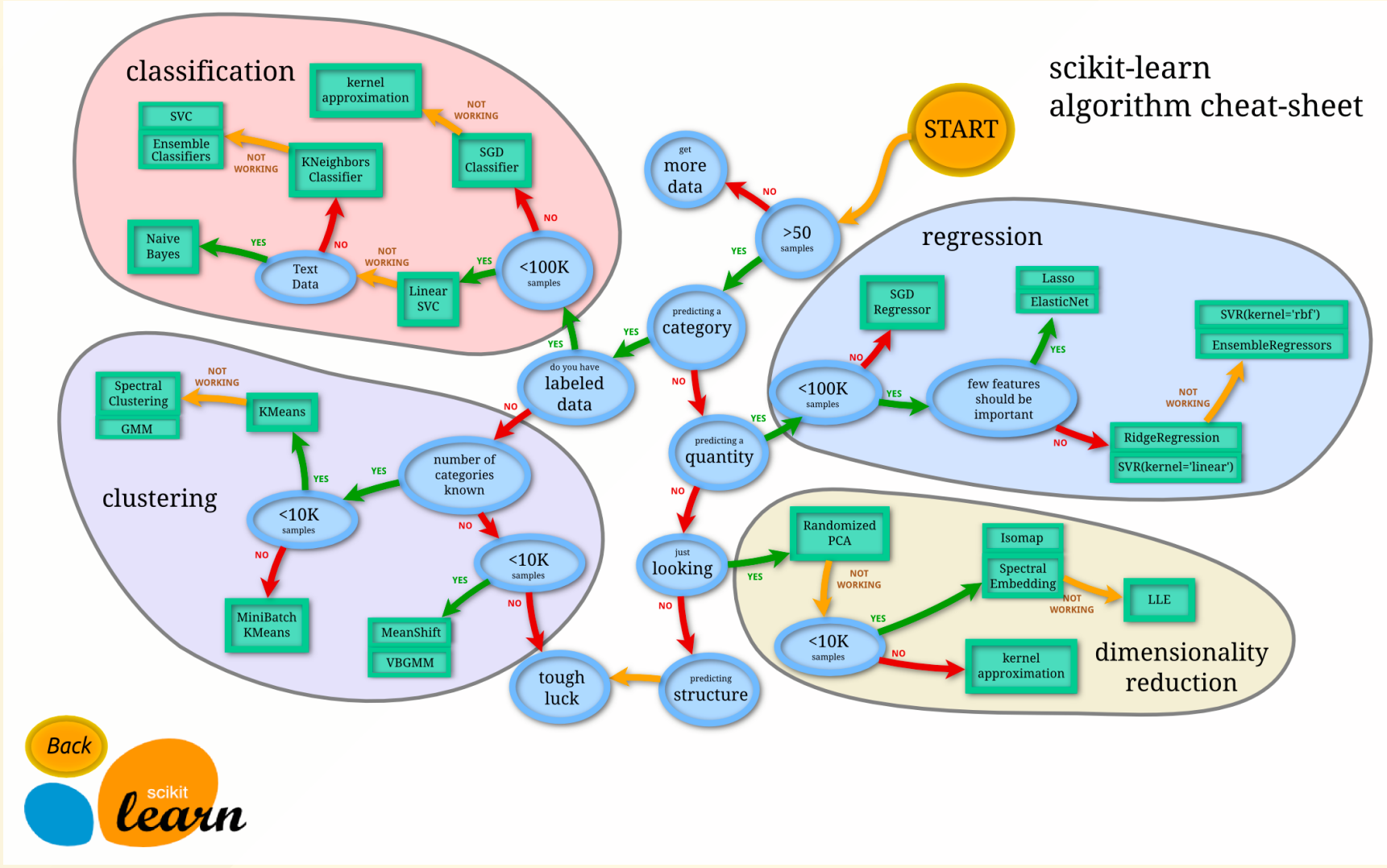
library

- math: `math`
- priority queue: `heapq`
- permutation, combinations, etc: `itertools`
- regular expression: `re`
- parser for command-line: `argparse`
- system command, path, etc: `sys, os`
- image processing: `pillow`
- web development: `django, flask`
- sql: `sqlite3`
- .....

data science

- array, matrix: `numpy`
- linear algebra, statistics: `scipy`
- symbolic calculation: `sympy`
- visualization: `matplotlib, seaborn`
- data manipulation and analysis: `pandas`
- machine learning: `scikit-learn`
- deep learning: `TensorFlow, PyTorch`

sklearn


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
YES
Linear SVC
<100K samples
NO
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
predicting a category
YES
NO
<100K samples
NO
YES
few features should be important
YES
NOT WORKING
RidgeRegression
NO
SVR(kernel='linear')
YES
do you have labeled data
NO
predicting a quantity
YES
NO
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
<10K samples
NO
NO
<10K samples
YES
NO
MiniBatch KMeans
MeanShift
VBGMM
just looking
YES
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
NO
tough luck
predicting structure
Back
scikit learn

More Resources:

- More cheatsheet: https://github.com/kailashahirwar/cheatsheets-ai
- Think Python: http://greenteapress.com/thinkpython2/thinkpython2.pdf

Thanks