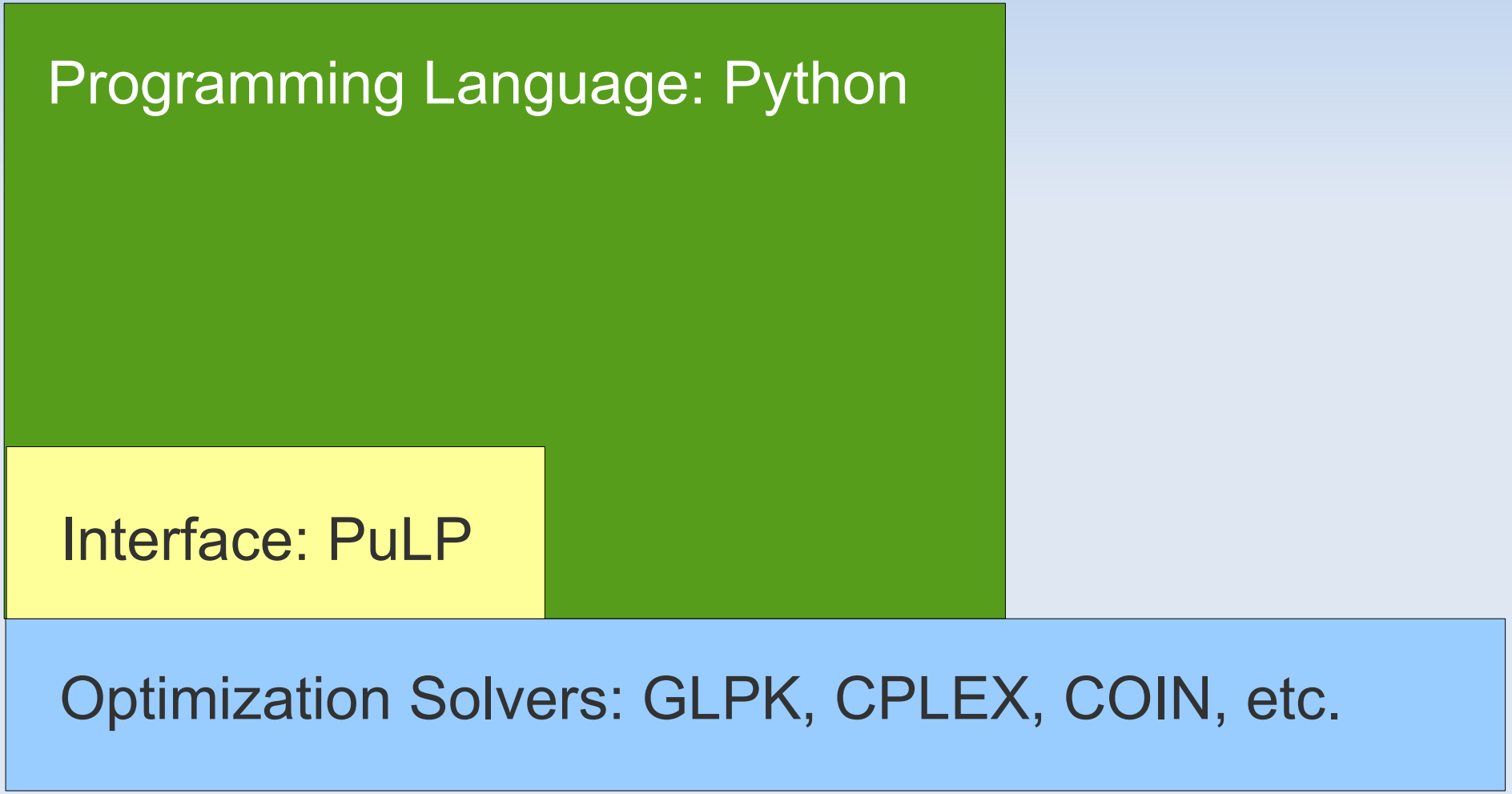
Tutorial: Python, PuLP & GLPK

- **Sucha Supittayapornpong**

Twitter: @Sucha

5 Mar. 2010

Architecture



Python

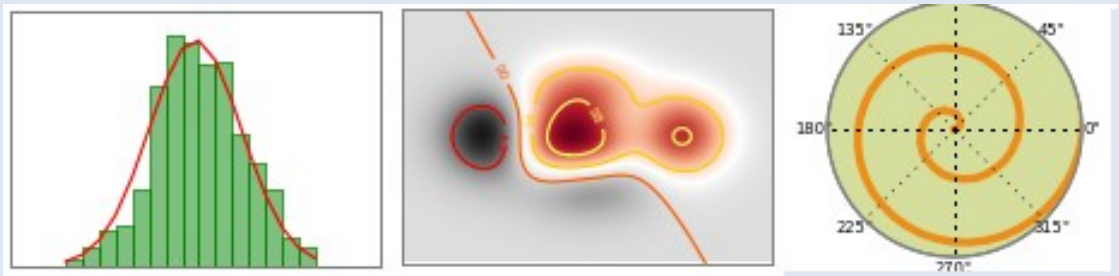
- Python is a programming language.
- Python runs on Windows, Linux/Unix, Mac OS X.
- Python is free to use.

python™

From: http://www.python.org/

Python: Features

- High-level data structures
  - Ex: list, tuple, dictionary
- Object-oriented
- Interpreter
- Standard library & Third party modules
  - Ex: pulp, numpy, matplotlib



From: http://code.google.com/p/pulp-or/ , http://numpy.scipy.org/ , http://matplotlib.sourceforge.net/

Python: Basic Data Type

- Integer (int)
  - Ex: 1, 2, 3, 5, 8, 13, 21
- Float (float)
  - Ex: 3.14, 2.71828
- Boolean (bool)
  - Ex: True, False
- String (str)
  - Ex: 'Python', ”Sucha”

Python: High-Level Data Type

- List (list): [d1, d2, …, dn]
  - Ex: [1, 3.14, True, 'a', [1], (2,3), {'B+':3.5}]
- Tuple (tuple): (d1, d2, ..., dn,)
  - Ex: (1, 3.14, True, 'a', [1], (2,3), {'B+':3.5})
- Dictionary (dict): { k1:v1 , k2:v2 , ..., kn:vn }
  - Ex: { 'A':4 , 'B+':3.5 , 3:'B' }
- Set (set): set([d1, d2, ..., dn])
  - Ex: set([4, 3.5, 'B'])

Python: Flow Control - If

- If statement

```
if boolean:
    command
elif boolean:
    command
else:
    command
```

Python: Loop - For, While

- For loop

  for *var* in *sequence*:

    command

- While loop

  while *boolean*:

    command

Python: List Comprehensions

- >>>[i for i in range(5)]

  → [0, 1, 2, 3, 4]

- >>>[i for i in range(5) if i <> 3]

  → [0, 1, 2, 4]

- >>>[(i, j) for i in range(3) for j in range(i)]

  → [(1,0), (2,0), (2,1)]

PuLP & GLPK

- PuLP is an LP modeler written in Python.
- PuLP can generate LP files, and calls solvers to solve linear problems.
- Supported solvers are
  GLPK, COIN, CPLEX, and GUROBI.

  http://code.google.com/p/pulp-or/
- The GLPK (GNU Linear Programming Kit) package is intended for solving
  large-scale linear programming (LP),
  mixed integer programming (MIP),
  and other related problems.

  http://www.gnu.org/software/glpk/

PuLP: Import Module

- Import module
  - >>>import pulp

    >>>pulp.pulpTestAll()

  - >>>from pulp import *

    >>>pulpTestAll()

Following slides assume the first import method.

PuLP: Create Decision Variables

- DV = pulp.LpVariable(*name_str*,
  lowbound,
  upbound,
  category)
- For *lowbound* and *upbound*, No bound → None .
- category ∈ { pulp.LpContinuous,
  pulp.LpInteger,
  pulp.LpBinary }
- Ex: x ∈ [0, ∞)
  x = pulp.LpVariable('Var X', 0, None, pulp.LpContinuous)

PuLP: Formulate Problem

- PB = pulp.LpProblem(*name_str*, *sense*)
- *sense* ∈{ pulp.LpMinimize, pulp.LpMaximize }
- Ex: maximization problem

```
prob = pulp.LpProblem('Benefit', pulp.LpMaximize)
```

PuLP: Add Objective Function

- PB += *linear_function*, *objective_name_str*
- *linear_function* is in the form of
  c1*DV1 + c2*DV2 + … + cn*DVn

- Ex: Cost: 2*DV1 – 1.5*DV2
  prob += 2*x1 – 1.5*x2, 'Cost'

PuLP: Add Constraints

- PB += *linear_constraint* , constraint_name_str
- *linear_constraint* is in the form of

  a1*DV1 + a2*DV2 + … + an*DVn == a0

  or a1*DV1 + a2*DV2 + … + an*DVn <= a0

  or a1*DV1 + a2*DV2 + … + an*DVn >= a0

- Ex: Con1: 5*DV1 + 6*DV2 <= 7

  prob += 5*x1 + 6*x2 <= 7, 'Con1'

  or

  prob += 2*x1 + 6*x2 <= 7 – 3*x1, 'Con1'

PuLP: Write .lp File

- PB.writeLP(*filename_str*)

- Ex: write to Benefit.lp
  prob.writeLP('Benefit.lp')

PuLP: Solve

- PB.solve() // Solved by COIN solver
- Ex: prob.solve()

- PB.solve(pulp.GLPK()) //Solved by GLPK solver
- Ex: prob.solve(pulp.GLPK())

PuLP: Results

- Check status: pulp.LpStatus[PB.status]
- Ex: pulp.LpStatus[prob.status]

- Optimal cost: pulp.value(PB.objective)
- Ex: pulp.value(prob.objective)

- Optimal solution: DV.varValue
- Ex: x1.varValue
  or
  pulp.value(x1)