Programming for Data Science – Introduction to Python

Henrik Boström
Prof. of Computer Science - Data Science Systems
Division of Software and Computer Systems
Department of Computer Science
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
bostromh@kth.se

Outline

Installing Python

Variables, numbers, strings, booleans and casting

Operators

Lists, tuples, sets and dictionaries

If statements, for and while loops

List comprehensions

Functions

Classes and objects

Modules

Input/output

Installing Python

- The offical Python website is `www.python.org`, where downloads, tutorials, community, etc. may be found
- A convenient way of installing Python together with a large number of packages (several to be used during the course) is to install Anaconda (`www.anaconda.com/download/`)
- Find some suitable IDE/working environment, e.g., PyCharm, PyDev, Jupyter, Emacs
- Note that the assignments have to be submitted in the form of Jupyter notebooks; see instructions in Canvas

Variables and numbers

- A variable is created when a value is assigned to it

```
v = 3.6
```

- There are three types of numbers; `int`, `float` and `complex`

```
i = 314
f = 3.14e2
z = 2+3j
```

- The type of a variable can be checked with `isinstance(...)`

```
b = isinstance(i,float)        # b = False
```

Strings

- Strings (`str`) are surrounded by single or double quotes

```
s = 'hello'                    # s = 'hello'
b = isinstance("i",str)        # b = True
```

Booleans

- The Boolean type (`bool`) consists of the values True and False

```
b1 = True                       # b1 = True
b2 = isinstance(b1,bool)        # b2 = True
b3 = isinstance("True",bool)    # b3 = False
```

Casting

- Casting using constructor functions; `int(...)`, `float(...)`, `str(...)`, `bool(...)`

```
i = int(3.14)              # i = 3
f = float(3)               # f = 3.0
s = str(3.14)              # s = "3.14"
f = float(s)               # f = 3.14
i = int(float("3.14"))     # i = 3

b = bool("hello")          # b = True
b = bool("")               # b = False
b = bool(5)                # b = True
b = bool(0.0)              # b = False
```

Operators

- Arithmetic operators; +, −, *, /, ** (exp.), // (floor div.), % (modulus)

```
v = 2.0 + 2**3              # v = 10.0
```

- Assignment operators; =, +=, −=, *=, /=

```
x = 12
x += 2                      # x = 14
```

- Comparison operators; ==, !=, >, <, >=, <=

```
b = (2.0 == 2)              # b = True
```

Operators (cont.)

- Logical operators; `and`, `or`, `not`

```
b = (1+1 == 2 and not 4>5)     # b = True
```

- Identity operators; `is`, `is not`

```
b = 2 is not 2.0               # b = True
b = 1+1 is 2                   # b = True
```

Operator precedence

Lowest precedence (least binding) to highest precedence (most binding) operators:

$=$

if - else

or

and

not

in, not in, is, is not, $<$, $<=$, $>$, $>=$, $!=$, $==$

$+$, -

*, /, //, %

**

$(\ldots)$

Lists

- Lists (indexed, ordered, changeable)

```
lang = ["Python","r","Julia"]
first = lang[0]             # first = "Python"
first_two = lang[0:2]       # first_two = ["Python","r"]
all_but_first = lang[1:]    # all_but_first = ["r","Julia"]
all_but_last = lang[:-1]    # all_but_last = ["Python","r"]
lang[1] = "R"               # lang = ["Python","R","Julia"]
lang += ["S"]               # lang = ["Python","R","Julia","S"]
l1 = ["a","b","b"]
len(l1)                     # 3
l1.count("b")               # 2
l1.append("c")              # l1 = ["a","b","b","c"]
l1.remove("b")              # l1 = ["a","b","c"]
l1.extend(["d","e"])        # l1 = ["a","b","c","d","e"]
l2 = [1,2,3]+[4,5]          # l2 = [1,2,3,4,5]
1 in l2                     # True
```

Tuples

- Tuples (indexed, ordered, items cannot be changed)

```
fixed = ("a","b","c")
f = fixed[1]                    # f = "b"
fixed[0] = "d"                  # Results in error
"c" in fixed                    # True
```

Sets

- Sets (not indexed, unordered, no duplicates)

```
s = {"a","b","b","c"}          # s = {"a","b","c"}
s.remove("a")                  # s = {"b","c"}
s.add("d")                     # s = {"b","c","d"}
s = s.union(set(lang))
s = s.difference(set(["b","c","d"]))
s = s.intersection(set(["Python","F#"]))
"S" in s                       # False
s < set(lang)                  # True
```

Dictionaries

- Dictionaries (indexed, unordered, changeable)

```
d = {"Python":1994,"R":1995,"Julia":2018}
y = d["R"]                    # y = 1995
d["S"] = 1976
list(d.keys())                # ["Python", "R", "Julia", "S"]
list(d.values())              # [1994, 1995, 2018, 1976]

t = d["T"]                    # Key error
t = d.get("T")                # t = None
t = d.get("T", 0)             # t = 0

d2 = {("a",1):500, ("b",2):250}
d2[("b",2)]                   # returns 250
```

If statements

- if statements (with `elif` and `else`)

```
if n>5:
   print("more than 5")
elif n == 5:                 # - elif not required,
   print("equal to 5")       # and multiple allowed
else:                        # - else not required
   print("less than 5")      # and most one allowed
```

For loops

- for loops

```
for i in range(3):              # Prints 0, 1, 2
   print(i)

for i in [1,2,3]:               # Prints 1, 2, 3
   print(i)

for i in "hello":               # Prints h, e, l, l, o
   print(i)

for e in enumerate(["a","b","c"]):
   print(e)                     # Prints (0,a), (1,b), (2,c)

for e in enumerate(["a","b","c"]):
   if e[0] % 2 == 0:            # Prints a, c
      print(e[1])
```

For loops (cont.)

- `for` loops (with `break` and `continue`)

```
for i in [1,2,3]:              # Prints 1
    if i % 2 == 0:
        break
    print(i)

for i in [1,2,3]:              # Prints 1,3
    if i % 2 == 0:
        continue
    print(i)
```

While loops

- `while` loops (with `break` and `continue`)

```
i = 1
while i < 4:                # Prints 1, 2, 3
    print(i)
    i += 1

i = 1
while i < 4:                # Prints 1
    if i % 2 == 0:
        break
    print(i)
    i += 1
```

While loops (cont.)

- `while` loops (with `break` and `continue`)

```
i = 1
while i < 4:
    if i % 2 == 0:
        continue
    print(i)
    i += 1
```

```
# Prints 1 and then
# enters infinite loop
```

List comprehensions

- Creating lists without for/while loops

```
nl = []
for la in lang:
    nl += [la.lower()]

# Equivalent (but more efficient):
nl = [la.lower() for la in lang]

# Include only items with multiple characters
nl = [la.lower() for la in lang if len(la) > 1]

# Convert items only with multiple characters
nl = [la.lower() if len(la) > 1 else la for la in lang]

# Generate a list with all characters
cs = [c for la in lang for c in la]
```

Functions

- functions (using `def` and `return`)

```
def add_one_and_print(a):
   a += 1
   print(a)
   return a
b = 1
c = add_one_and_print(b)       # 2 is printed and c = 2
print(b)                       # 1 is printed

def add_two_to_second(l1):
   l1[1] += 2
l = [1,2,3,4,5]
r = add_two_to_second(l)       # Note: l = [1,4,3,4,5]
r is None                      # True
```

Functions (cont.)

- functions with default argument values

```
def diff(a=10,b=20):
    return a-b

d0 = diff()                    # d0 = -10
d1 = diff(5,6)                 # d1 = -1
d2 = diff(5)                   # d2 = -15
d3 = diff(b=5)                 # d3 = 5
d4 = diff(b=2,a=3)             # d4 = 1
```

Lambda functions

- Lambda functions = anonymous functions with one expression

```
r = (lambda x: x+1)(5)          # r = 6

f = lambda x,y: x+y
sum = f(2,3)                    # sum = 5

def deriv(f,x,h):
   return (f(x+h)-f(x))/h

deriv(lambda x: x**2,8,1e-10) # 16.000001323845936
```

Classes and objects

- Class definitions (using `class`)

```
class DSLang:
    def __init__(self, name, year):
        self.name = name
        self.year = year

l1 = DSLang("Python",1994)
l2 = DSLang("Julia",2018)
print(l1.name)                # Prints Python
```

Classes and Objects (cont.)

- Methods

```
class DSLang:
    def __init__(self, name, year):
        self.name = name
        self.year = year

    def age(self,current_year):
        return current_year-self.year

l2 = DSLang("Julia",2018)
print(l2.age(2019))          # Prints 1
```

Classes and objects (cont.)

- Special methods

```
class Super:
   def __init__(self, age):
      self.age = age
   def __str__(self):
      return "My age is: "+str(self.age)
   def __eq__(self,value):
      return self.age == value
   def __len__(self):
      return self.age

o = Super(5)
print(o)                           # My age is: 5
o == 5                             # True
```

Classes and objects (cont.)

- Inheritance

```
class Sub(Super):
    def __init__(self,age=3):
        self.age = age

s = Sub()
print(s)                            # My age is: 3
len(s)                              # 3
```

Modules

- Define a module by placing your code in a file, named with the extension .py

```python
# In the file my_definitions.py

class DSLang:
    def __init__(self, name, year):
        self.name = name
        self.year = year
```

Modules (cont.)

- Import a module and use its definitions

```
import my_definitions
lo = my_definitions.DSLang("R",1995)

import my_definitions as md
lo = md.DSLang("R",1995)

from my_definitions import DSLang
lo = DSLang("R",1995)
```

Modules (cont.)

- Reloading a module (after having edited its definitions)

```
from importlib import reload

reload(my_definitions)
```

Input/output

- Write to standard output

```
print("R",1995)                         # R 1995
print("N:{} Y:{}".format("R",1995))     # N: R Y: 1995
print("F: {:.2f}".format(31.41592))     # F: 31.42
print("F: {:.4f}".format(31.41592))     # F: 31.4159
print("E: {:.4e}".format(31.41592))     # E: 3.1416e+01
print("S: {1:10.2f}".format(1,2,3))     # S:       2.00
```

- Read from standard input

```
s = input()                             # s will be assigned
                                        # a string
```

Input/output (cont.)

- Write to files

```
f = open("temp.txt","w")    # Opens file for (over-)writing
result = [1,2,3]
f.write(str(result))        # Only strings can be written
f.close()

f = open("temp.txt","a")    # Opens file for appending text
f.write("Bye!\n")
f.close()
```

Summary

- We have covered a large part (but not all) of the syntax and semantics of Python (check the documentation for additional features)
- It should be noted that Python has primarily been developed for ease-of-use rather than with efficiency in mind
- Together with libraries, such as NumPy and pandas (covered in the next lecture), it has become a standard tool for data scientists