Applied Statistical Methods

Introduction to Python - Part III

Xuemao Zhang
East Stroudsburg University

January 27, 2023

Outline

- Python functions
- lambda function
- Flow Control and Loops
    - Python `if ... else`
    - Python `for` loops
    - Python `while` loops

Python functions

- A function takes a list of argument values, performs a computation with those values, and returns a single result. Python gives you many built-in functions.
  - see Python Built-in Functions https://docs.python.org/3/library/functions.html

```
print(abs(-2))
```

```
## 2
```

```
print(round(3.1415926,3))
```

```
## 3.142
```

```
x = str(23.5)
print(type(x))
```

```
## <class 'str'>
```

```
print(x)
```

```
## 23.5
```

```
print(any([True,False]))
```

```
## True
```

Python functions

- We can also create our own functions. These functions are called user-defined functions.
- Functions are declared using the **def** keyword, and the value produced is returned using the **return** keyword. Consider a simple function which returns the square of the input, $y = x^2$.
  - colon : is used to represent an indented block. It is not for slicing.
  - Python uses **indentation** to indicate a block of code.
  - The number of spaces is up to you as a programmer, but it has to be at least one.

```
def square(x):
  return x**2

x = 2
y = square(x) # Call the function
print(x,y)
```

```
## 2 4
```

Python functions

```python
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
```

```
## Emil Refsnes
```

```python
my_function("Tobias")
```

```
## Tobias Refsnes
```

```python
my_function("Linus")
```

```
## Linus Refsnes
```

Python functions

- Function can also be defined using NumPy arrays

```
import numpy as np
def L2_norm(x,y):
  d=x-y
  return np.sqrt(np.dot(d,d))
x = np.random.randn(10)
y = np.random.randn(10)
z = L2_norm(x,y) #call the function
print(x,y)
```

```
## [ 1.01847198  1.66141342 -1.44376514  0.25232728 -1.09006839 -0.95931097
##   0.21122656 -2.15823397 -0.06195763 -0.28505463] [-0.07743353 -1.408810
##  -0.03832     0.07034394 -0.88866189 -0.71279391]
```

```
print("The L2 distance is ",z)
```

```
## The L2 distance is  4.754860326674871
```

- `random.randn` return a sample (or samples) from the "standard normal" distribution. https://numpy.org/doc/stable/reference/random/generated/numpy.random.randn.html

lambda function

- A `lambda function` is a small anonymous function.
- A lambda function can take any number of arguments, but can only have one expression.
- Syntax: `lambda arguments: expression`

```
x = lambda a, b : a * b
print(x(5, 6))
```

```
## 30
```

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
## 13
```

lambda function

- The power of lambda is better shown when you use them as an anonymous function inside another function.

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)
#it is a function returned by the lambda function
print(mydoubler(11))
```

```
## 22
```

```
mytripler = myfunc(3)
print(mytripler(11))
```

```
## 33
```

Python `if ... else`

- Comparison Operators are used to evaluate to True or False depending on input condition.
- An `if statement` is written by using the **if** keyword.
  - Python relies on indentation to define scope in the code. Other programming languages often use curly-brackets for this purpose.
- `if` keyword

```
if logical:
   Code to run if logical True
```

```python
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
## b is greater than a
```

Python if ... else

- else keyword

```
if logical:
    Code to run if logical True
else:
    Code to run if logical False
```

```python
name = 'Debora'
if name == 'George':
  print('Hi, George.')
else:
  print('You are not George')
```

```
## You are not George
```

Python `if ... else`

- The **elif** keyword is pythons way of saying “if the previous conditions were not true, then try this condition” or `else if`.

```python
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:  #no indentation
  print("a and b are equal")
```

```
## a and b are equal
```

Python if ... else

- The **else** keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
## a is greater than b
```

Python if ... else

- One more example

```python
x = 5
if x<5:
    x+=1
elif x>5:
    x-=1
else:
    x=x**2
print(x)
```

```
## 25
```

Python `if ... else`

- Short Hand If

```
if a > b: print("a is greater than b")
```

```
## a is greater than b
```

- Short Hand If . . . Else

```
a = 2
b = 330
print("A") if a > b else print("B")
```

```
## B
```

- This technique is known as Ternary Operators, or Conditional Expressions.
- You can also have multiple else statements on the same line:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

```
## =
```

Python `if ... else`

- The `and` keyword is used to combine conditional statements

```python
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

```
## Both conditions are True
```

- The `or` keyword is used to combine conditional statements

```python
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
## At least one of the conditions is True
```

Python if ... else

- Nested If: You can have if statements inside if statements, this is called nested if statements.

```
x = 9
if x > 10:
  print("Above ten,")
  if x > 20:  # 'x>10' is True; indentation
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Python `if ... else`

```
x = 9
if x > 10:
  print("Above ten,")
  if x > 20:  # 'x>10' is True; indentation
    print("and also above 20!")
  else:
    print("but not above 20.")
else:
  print("not greater than 10")
```

```
## not greater than 10
```

Python `if ... else`

- The **pass** Statement: if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
  pass
```

Python for loops

- A for loop is used for iterating over a sequence (that is either a range, list, a tuple, a dictionary, a set, or a string).

```
for item in iterable:
  Code to run
```

- The range() function allows you to iterate over a sequence of numbers. It starts from 0, increments by 1, and stops before a specified number

```
for i in range(4):
  print(i)
```

```
## 0
## 1
## 2
## 3
```

Python for loops

```python
for i in range(1, 5, 1):
 print(i)
```

```
## 1
## 2
## 3
## 4
```

```python
for i in range(1, 10, 2):
 print(i)
```

```
## 1
## 3
## 5
## 7
## 9
```

Python `for` loops

- Looping through a list

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
## apple
## banana
## cherry
```

- Looping through a string

```
for x in "banana":
  print(x)
```

```
## b
## a
## n
## a
## n
## a
```

Python `for` loops

- The `break` statement: with the `break` statement we can stop the loop before it has looped through all the items

```python
for i in [1, 2, 3, 4, 5]:
  if i == 3:
    break
  else:
    print(i)
```

```
## 1
## 2
```

Python for loops

- The continue statement: with the continue statement we can stop the current iteration of the loop, and continue with the next

```python
for i in [1, 2, 3, 4, 5]:
  if i == 3:
    continue
  else:
    print(i)
```

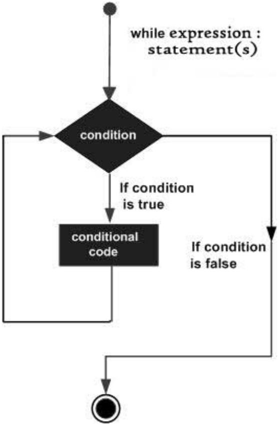
```
## 1
## 2
## 4
## 5
```

```python
for i in [1, 2, 3, 4, 5]:
  if i == 3:
    continue
  print(i)
```

```
## 1
## 2
## 4
## 5
```

Python `while` loops

- With the `while` loop we can execute a set of statements as long as a condition is true.



Python while loops

```
a=5
while a<10:
  print(a)
  a+=1
```

```
## 5
## 6
## 7
## 8
## 9
```

```
print("Out of Loop")
```

```
## Out of Loop
```

- while loops should generally be avoided when for loops are sufficient. However, there are situations, for example the number of iterations required is not known in advance, where no for loop equivalent exists.

License