POL 42340 Programming for Social Scientists: What you will know after each class

Johan A. Elkink

20 October 2020

Contents

This document provides an overview of all the Python code you will learn in this course, by week. Don't worry if you do not know most of this code yet. After each week, you will know the code for that specific week. If you still have questions, ask on Slack, in class, or search the web for more information about that command in Python.

For the preparation of this document, extensive use was made of David M. Beazley, *Python: Essential Reference*, 4th edition (2009).

Week 1: Introduction

```
print("Hello, World!")
```

```
Hello, World!
```

Using Python as calculator

```
5 + 30 * 3
```

```
95
```

```
2 ** 3
```

```
8
```

```
5 / 2
```

```
2.5
```

```
5.0 / 2
```

```
2.5
```

```
5 // 2
```

```
2
```

```
5.0 // 2
```

```
2.0
```

Week 2: Variables and functions

Variables

```
a = 4

type(a)
```

```
<class 'int'>
```

```
b = 4.0

type(b)
```

```
<class 'float'>
```

```
s = "Pythons are scary!"

type(s)
```

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

Functions

```
def add_five(base):
  return base + 5

add_five(10)
```

```
15
```

```
a = 4
add_five(a)
```

```
9
```

Using default parameter values

```
def add(base, addition = 1):
  return base + addition

add(5, 3)
```

```
8
```

```
add(5)
```

```
6
```

You will need to pay particular attention to where a variable is valid, the *scope* of the variable, so that you do not get confused by code like this:

```
x = 3

def subtract_five(base):
  x = 5
  return base - x

subtract_five(x)
```

```
-2
```

Random numbers

```
import random

a = random.randint(1, 6)
a
```

```
3
```

```
type(a)
```

```
<class 'int'>
```

```
b = random.random()
b
```

```
0.8391695100732793
```

```
type(b)
```

```
<class 'float'>
```

Week 3: Conditions and flow

```
age_Mark = 43
age_Toby = 30

if age_Mark < age_Toby:
  print("Toby is older than Mark")
else:
  print("Mark is older than Toby")
```

```
Mark is older than Toby
```

```
product = "book"
price = 18.95

if product != "book":
  print("The product is not a book")
else:
  print("The product is a book")
```

```
The product is a book
```

```
if product == "book" and price > 15:
  print("The product is an expensive book")
elif product == "book":
  print("The product is a cheap book")
else:
  print("The product is not a book")
```

```
The product is an expensive book
```

Assigning a comparison to a variable

```
is_cheap_book = product == "book" and price <= 15
is_cheap_book
```

```
False
```

```
type(is_cheap_book)
```

```
<class 'bool'>
```

While loops

```
loan_amount = 100.0
initial_amount = loan_amount
interest_rate = 0.02
annuity = 5.0
payments = 0

while loan_amount > 0:
  loan_amount = loan_amount * (1 + interest_rate) - annuity
  payments += 1

print("With a loan of " + format(initial_amount, ".2f") + \
" and an interest rate of " + format(interest_rate, ".2f") + \
", it takes " + format(payments, "d") + \
" payments of " + format(annuity, ".2f") + " to pay back the loan.")
```

```
With a loan of 100.00 and an interest rate of 0.02, it takes 26 payments of 5.00 to pay back the loan.
```

```
print("You will have paid " + format(payments * annuity + loan_amount, ".2f") + " in total.")
```

```
You will have paid 128.99 in total.
```

Week 4: Loops and lists

Lists

```
names = [ "Dave", "Mark", "Ann", "Phil" ]
names[2]
```

```
'Ann'
```

```
names[1:3]
```

```
['Mark', 'Ann']
```

```
names[2] = "Jos"
names
```

```
['Dave', 'Mark', 'Jos', 'Phil']
```

```
names.append("Fredrik")
names
```

```
['Dave', 'Mark', 'Jos', 'Phil', 'Fredrik']
```

```
names.insert(2, "Sam")
names
```

```
['Dave', 'Mark', 'Sam', 'Jos', 'Phil', 'Fredrik']
```

```
messy_list = [1,"Dave",3.14, ["Mark", 7, 9, [100,101]], 10]
messy_list[3]
```

```
['Mark', 7, 9, [100, 101]]
```

```python
messy_list[3][3]
```

```
[100, 101]
```

Loops

```python
range(5)
```

```
range(0, 5)
```

```python
for i in range(5):
  print("This is iteration " + format(i, "d"))
```

```
This is iteration 0
This is iteration 1
This is iteration 2
This is iteration 3
This is iteration 4
```

```python
names = [ "Dave", "Mark", "Ann", "Phil" ]
for name in names:
  print(name + " is in the list")
```

```
Dave is in the list
Mark is in the list
Ann is in the list
Phil is in the list
```

Dictionaries

```python
stock_prices = {
  "GOOG": 240.10,
  "AAPL": 123.50,
  "IBM": 87.50,
  "MSFT": 53.13
}

if "AAPL" in stock_prices:
  print(stock_prices["AAPL"])
else:
  print("AAPL not found")
```

```
123.5
```

```python
for stock in stock_prices:
  print("The price of " + stock + " is " + format(stock_prices[stock], ".2f"))
```

```
The price of GOOG is 240.10
The price of AAPL is 123.50
The price of IBM is 87.50
The price of MSFT is 53.13
```

Week 5: Working with text

```
s = "Hello, World!"

len(s)
```

```
13
```

```
s[:3]
```

```
'Hel'
```

```
s[7:12]
```

```
'World'
```

```
s + " It's a beautiful day!"
```

```
"Hello, World! It's a beautiful day!"
```

```
'"' + s + '", he said.'
```

```
'"Hello, World!", he said.'
```

Look very carefully at the use of quotation marks on that last one!

Conversion between numbers and strings

```
s = "23"
s.isnumeric()
```

```
True
```

```
a = int(s)
a
```

```
23
```

```
type(s)
```

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

```
type(a)
```

```
<class 'int'>
```

```
s = "24.1093"
s.isnumeric()
```

```
False
```

```
a = float(s)
a
```

```
24.1093
```

```
type(s)
```

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

```
type(a)
```

```
<class 'float'>
```

```
format(a, ".2f")
```

```
'24.11'
```

Input of text from the user

```
import random

number = input("How many random numbers do you want? ")

if number.isnumeric():
  result = []
  for i in range(int(number)):
    result.append(random.random())
  print(result)
else:
  print("ERROR: Must enter a number!")
```

Week 6: Object-oriented programming

```
class Circle(object):

  def __init__(self, name = "", radius = 0.0, x = 0.0, y = 0.0):
    self.radius = radius
    self.x = x
    self.y = y
    self.name = name

  def set_center(self, x, y):
    self.x = x
    self.y = y

  def set_radius(self, r):
    self.radius = r

  def set_name(self, s):
    self.name = s

  def blow_up(self, multiplier = 1.1):
    self.radius *= multiplier

  def __repr__(self):
    return self.name + ": " + format(self.radius, ".2f") + \
                       " at (" + format(self.x, ".2f") + \
                       "," + format(self.y, ".2f") + ")"

c = Circle()

c
```

```
: 0.00 at (0.00,0.00)
```

```
c.set_name("Blue ball")
c.set_radius(2.0)
c.set_center(-1, 3)

c
```

```
Blue ball: 2.00 at (-1.00,3.00)
```

```
c.blow_up(1.5)

c
```

```
Blue ball: 3.00 at (-1.00,3.00)
```

```
c.blow_up()

c
```

```
Blue ball: 3.30 at (-1.00,3.00)
```

```
d = Circle("Red ball", 1.5, 1, 2)

d
```

```
Red ball: 1.50 at (1.00,2.00)
```

```
balls = [c, d]
balls
```

```
[Blue ball: 3.30 at (-1.00,3.00), Red ball: 1.50 at (1.00,2.00)]
```

Week 7: Working with files

An input file

```
line_number = 0

for line in open("POL42340_Autumn_2020_overview_python_code.Rmd"):

  print(line.strip())

  line_number += 1
  if line_number == 2:
    break
```

```
---
title: "POL 42340 Programming for Social Scientists: What you will know after each class"
```

An output file

```
log_file = open("test_log_file.txt", "w")

for i in range(4):
  log_file.write("Log line " + format(i, "2d") + "\n")
```

```
12
12
12
12
```

```
log_file.close()

for line in open("test_log_file.txt"):
  print(line.strip())
```

```
Log line  0
Log line  1
Log line  2
Log line  3
```

```
import os

os.remove("test_log_file.txt")
```

Week 8: Handling exceptions (and tuples)

Catching errors

```
try:
  for line in open("test.txt", "r"):
    print(line)
except IOError as e:
  print("Could not open file! " + e.strerror)
```

```
Could not open file! No such file or directory
```

Tuples

```

book_hhgg = ("The Hitchhiker's Guide to the Galaxy", 18.34)
book_tr = ("The Real Donald Trump", 2.50)
book_cap = ("The Great Capsize", 99.50)

books = [book_hhgg, book_tr, book_cap]

books
```

```
[("The Hitchhiker's Guide to the Galaxy", 18.34), ('The Real Donald Trump', 2.5), ('The Great Capsize',
```

```
books[0]
```

```
("The Hitchhiker's Guide to the Galaxy", 18.34)
```

```
books[0][0]
```

```
"The Hitchhiker's Guide to the Galaxy"
```

```python
type(books)
```

```
<class 'list'>
```

```python
type(books[0])
```

```
<class 'tuple'>
```

```python
type(books[0][0])
```

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

```python
title, price = books[0]
print("The book " + title + " has a price of " + format(price, ".2f"))
```

```
The book The Hitchhiker's Guide to the Galaxy has a price of 18.34
```

```python
total_cost = 0
for book in books:
  total_cost += book[1]
print("Total price: " + format(total_cost, ".2f"))
```

```
Total price: 120.34
```

```python
total_cost = 0
for title, price in books:
  total_cost += price
print("Total price: " + format(total_cost, ".2f"))
```

```
Total price: 120.34
```

```python
budget = 100
total_price = 0
i = 0
while total_price + books[i][1] < budget:
  total_price += books[i][1]
  i += 1
print("With my budget I end up buying " + format(i, "d") + " books, spending " \
    + format(total_price, ".2f"))
```

```
With my budget I end up buying 2 books, spending 20.84
```

Week 9: Working with libraries and GUIs

No specific code, but a general strengthening of your ability to search for documentation and use Python libraries.

Week 10: Computer simulations and visualisation

No specific code, but a general strengthening of your ability to search for documentation and use Python libraries.

And the rest

Some topics not covered here, that you might want to read up on later:

- Recursion
- Generators
- Coroutines
- Lambdas
- Unit testing
- The Python Library
- Python debugger