202503_Python-Gapminder_Day1_L0-1-2

March 4, 2025

0.1 Presenter notes with Carpentries Python Gapminder training

Martijn Wehrens, m.wehrens@uva.nl, 2025-03-04

These notes are intended to use as reference during the Carpentries Workshop for Python using Gapminder, during which live coding is used to teach people Python.

Installation notes to distribute among participants

What's needed

- Jupyter notebook (available online, see below).
- Dataset (avialable online, see below).

Details During this course, we'll be using a Jupyter notebook tool that's hosted online: - https://jupyter.org/try-jupyter/lab/ which is hosted by Jupyter.org.

A link to the Gapminder dataset is available on the Carpentries summary and setup page, here's a direct link: - https://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

Both of these can be installed during the workshop.

We'll also highlight tools people can use during the workshop.

Open considerations

- Should we allot some time to help people install their own software? Or maybe just have people come by later? Think about this.

1 Lesson 1: Running and quitting

(Content to be discussed.)

2 Lesson 2: Variables and assignments

2.0.1 Variables

- "names for values" / MW: a name in order to access data that's stored
- rules for names:
  - only letters, digits, underscore
  - cannot start with digit
    - starting w/ underscore also special
  - case sensitive
- use meaningful names

[2]:
```
age = 42 # the = sign is used to assign value to name on the left
first_name = 'Ahmed'
```

[3]:
```
# Print values:
print(first_name, 'is', age, 'years old')
    # Built-in "function" to print things as text
    # Call function by using name
    # Provide values within round brackets
    # additional strings provided between quotes
    # values passed = arguments

# Also, "print"
    # provides space
    # newline at end
```

```
Ahmed is 42 years old
```

[4]:
```
# Parameters must be defined before use

# Check typos also

print(last_name) # will throw error
    # Check out the (last line) of error message
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 5
      1 # Parameters must be defined before use
      2
      3 # Check typos also
----> 5 print(last_name)

NameError: name 'last_name' is not defined
```

Order of execution

- Order of execution
- Variables persist after execution

Example below

[6]:
```
print(myval) # won't work unless below executed

# To reset:
# Kernel -> Restart & Run All
```

```
1
```

[ ]:
```
myval = 1
```

2.0.2 Calculations

[ ]:
```
age = age + 3 # use variables in calculations, will use value

print('Age in three years:', age)
```

2.0.3 Using indices

- String is a series of symbols in specific order
  - Each position <-> number
    - called "index"
  - Indices start at 0!!!!
  - Use square brackets to access an element

[7]:
```
atom_name = 'helium'
print(atom_name[0])
```

```
h
```

[9]:
```
# Slices
# - part of string = substring
# - item in list = element
# - slice = part of "list-like thing"
# - [start:stop]
#       - start = first thing (0 = 1st)
#       - stop = right after last
#          - stop-start = length substring
# - slice doesn't edit original, but copy of selected part returned
atom_name='sodium'
print(atom_name[0:3])
```

```
sod
```

[ ]:
```
length_word_helium = len('helium')
print(length_word_helium)

print(len('helium'))
    # nested function
    # similar math
    # evaluated inside out
```

2.0.4 Remarks naming

- Case-sensitive
  - conventions about how to code
    - use lower case names
- Use meaningful names
  - python doesn't care
  - but code readability very important
    - make other people understand
    - other people = your future self!!!

[10]:
```
flabadab = 42
ewr_422_yY = 'Ahmed'
print(ewr_422_yY, 'is', flabadab, 'years old')
```

```
Ahmed is 42 years old
```

2.1 Exercises

1. Order of things

A Fill the table showing the values of the variables in this program after each statement is executed.

```
# Command  # Value of x   # Value of y   # Value of swap #
x = 1.0    #              #              #               #
y = 3.0    #              #              #               #
swap = x   #              #              #               #
x = y      #              #              #               #
y = swap   #              #              #               #
```

B What is the final value of position in the program below? (Try to predict the value without running the program, then check your prediction.)

```
initial = 'left'
position = initial
initial = 'right'
```

"Challenge" If you assign a = 123, what happens if you try to get the second digit of a via a[1]?

Naming Which is a better variable name, m, min, or minutes? Why? Hint: think about which code you would rather inherit from someone who is leaving the lab:

```
ts = m * 60 + s
tot_sec = min * 60 + sec
total_seconds = minutes * 60 + seconds
```

Slicing

A What does the following program print?

```
atom_name = 'carbon'
print('atom_name[1:3] is:', atom_name[1:3])
```

B

Given the following string:

```
species_name = "Acacia buxifolia"
```

what would these expressions return?

```
species_name[2:8]
species_name[11:] (without a value after the colon)
species_name[:4] (without a value before the colon)
species_name[:] (just a colon)
species_name[11:-3]
species_name[-5:-3]
```

What happens when you choose a stop value which is out of range? (i.e., try species_name[0:20] or species_name[:103])

Additional exercises for fast participants (by bioDSC) What's happening here:

```
# use Google or chatGPT if you don't know the answers
# if you're at the end, try to play around more

greetings_strings = ['hello', 'bye', 'later']
print(greetings_strings[0])
print(greetings_strings[1][2])
    # (list of lists)
```

```
print([greetings_strings[idx] for idx in [0, 2]])
print([greetings_strings[0][idx] for idx in [0, 2, 4]])
print([greetings_strings[0][idx] for idx in range(0,4,2)])
    # using a loop (will be covered later) / list comprehension

square_values = [number**2 for number in range(10)]
square_values_string = [str(number**2) for number in range(10)]
print(square_values)
print(square_values_string)
    # types will be the topic of the next lesson

species_name = "Acacia buxifolia"
print("".join([species_name[i] for i in range(10, 2, -1)]))
    # convenient command when working with strings

print(species_name.replace('Aca', 'Bole'))
    # another convenient command

# Exercise:
list_of_species = ['','','','']
list_of_species = ['Homo sapiens', 'Escherichia Coli', 'Pan troglodytes', 'Canis lupus', 'Felis
# Create a new lists, where you remove all letters 'e'
```