WORKSHOP
PYTHON

22th October 2018

Senne Deproost
sdeproos@infogroep.be
@zenne

REQUIREMENTS

- Python 3.X
- CMD or BASH
- Text editor

DURING THE WORKSHOP

- Execute “>>>” statements
- QUESTION? ==> ASK
- Slides available at `seminars.infogroep.be`


AFTER THE WORKSHOP...

BACKGROUND

- Dynamic language
- Object Oriented
- Runs on Python Virtual Machine
- Easy to use, reads like english

USE CASES

- Big data analysis
- Machine learning
- Prototyping
- Introduction to writing code

FIRST PROGRAM

- Open Python with command `python` in prompt
- After “>>>”, type:

```
print(“Hello World!”)
```

- Hit enter

TWO VERSIONS?

PYTHON 2.X

- Older version
- Small syntax differences
- Unfortunately, still widely used

```
print ”Hello World!”
```

PYTHON 3.X

- Newer version
- Syntax compatible with version 2 in most cases
- Conversion via 2to3
- F A S T E R

```
print(”Hello World!”)
```

ANYONE STILL VERSION 2!?

GOOD JOB

VARIABLES

- `>>> x = 5`
- No type mention needed, can be deducted from statement
- `>>> a = b = c = 42`

- `>>> spam = “eggs”`
- `>>> foo = False`

TYPES

- `>>> variable = 4.36`
- `>>> type(variable)`

- 5 standard types in python
- `Number(Int, Float, ...), String, List, Tuple, Dictionary`

- Find all methods of type/object: `dir`
- `>>> dir("Testje")`

DATASTRUCTURES

- Tuple
- List
- Dictionary
- ...

TUPLE

- Immutable
- ( )

- `>>> a_tuple = (1, 2, 3)`
- `>>> a_tuple[0]`
- `>>> a_tuple[0] = “eggs”`
- `>>> a_tuple`

LIST

- Mutable
- `[  ]`

- `>>> a_list = [1, 2, 3]`
- `>>> a_list[0]`
- `>>> a_list[0] = “eggs”`
- `>>> a_list`

LIST

- `>>> a_list.append(“Monty”)`
- `>>> a_list.append(“Python”)`
- `>>> del(a_list[1])`
- `>>> a_list`

- `>>> a_list[1:3]`
- `>>> a_list[2:3]`

DICTIONARY

- Key-Value pairs
- `>>> Contact_list = {“John Travolta”: 478901245, “Tom Javolta”: 23423423}`
- `>>> Contact_list[“Tom Javolta”]`
- `>>> Contact_list[“Simon Diaz”] = 2345345122`
- `>>> Contact_list`

DICTIONARY

- `>>> Contact_list.keys()`
- `>>> Contact_list.values()`

- `>>> Contact_list.clear()`
- `>>> Contact_list`

MATH OPERATORS

- `>>> 1 + 2`
- `>>> 4 / 2`
- `>>> 11 % 2`

MATH OPERATORS

- `>>> [“ha”] * 3`
- `>>> [34, 12] + [11]`

LOGIC OPERATORS

- and, or, not
- <, >, <=, >=
- ==, !=

- Booleans: True, False

LOGIC OPERATORS

- >>> “Pizzahut” > “Domino’s Pizza”
- >>> [1, 2, 3] > [1, 1, 1]

STRINGS

- “This is a string”, ‘This is also a string!’
- >>> sentence = “I am Liam”
- >>> sentence[2]
- >>> sentence[5:9]

STRINGS

- >>> “Mark” == “Marc”
- >>> “Mark” != “Marc”

- >>> long_sentence = “I believe this will be the longest sentence you have to type during this workshop.”
- >>> len(long_sentence)
- >>> “z” in long_sentence

STRINGS

- >>> “la” * 3
- >>> “Info” + “groep”

FUNCTIONS

- Work with code blocks: TAB->
-
```
>>> def hello(name):
...     print(“Hello, ” + name)
```

CONTROL FLOW

```
>>> if(1 < 4):
...     print(“Alright!”)
```

CONTROL FLOW

```
>>> if(42 < 4):
      print(“Alright!”)
    else:
      print(“Error!”)
```

CONTROL FLOW

```
>>> if(42 < 4):
      print(“Alright!”)
    elif(3 + 2 =! 4):
      print(“Good job!”)
    else:
      print(“Error!”)
```

LOOPS

```
>>> n = 0
>>> while(n <= 30):
      print(“Iteration ” + str(n))
      n = n+1
```

LOOPS

```
>>> for number in range(42):
      print(number)
```

BREAK

- Stop a loop from being executed
-
```
>>> for number in range(42):
        if (number > 23):
            break
        else:
            print(number)
```

PASS

- Null operator, replacing code to be written
-
```
>>> if True:
        pass
        print(“You... shall not... pass!!!”)
```

COMMENTS

- # This will be a comment
- ‘’’ And all of this
  will also be
  a comment ‘’’

EXECUTE PYTHON FILE

```
python file.py
```

INPUT

```
x = input('Enter your name: ')
print('Hello, ' + x)
```

CLASSES

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)
```

CLASSES

```
print(p1.name)
print(p1.age)


p1.myfunc()
```

PIP

- Python Package Manager
- Install libraries globally or local in python environment
- `pip`
- `pip list`

PIP

- Install multiple libraries with require file
- `pip install -r requirements.txt`

IMPORT

- `pip install pygame`
- Open text editor

IMPORT

```
import pygame

pygame.init()
width = 350
height = 200

#make the pygame window
pygame.display.set_mode((width, height ) )

running = True

while (running):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
```

WEATHER SERVER

- requirements.txt

```
flask
weather-api
```

WEATHER SERVER

- run_server.sh

```
FLASK_APP=server.py flask run
```

WEATHER SERVER

- forecast.py

```
from weather import Weather, Unit
weather = Weather(unit=Unit.CELSIUS)
city = "Borchtlombeek city"

location = weather.lookup_by_location(city)
forecasts = location.forecast
text = "Today, "

forecast = forecasts[0]
text = text + forecast.date + ", the weather in " + city +  " is " +
forecast.text + ", with max " + forecast.high + "°C and min " + forecast.low +
"°C."
```

WEATHER SERVER

- server.py

```
from flask import Flask
import forecast as f

app = Flask(__name__)

@app.route("/")
def hello():
    return f.text
```

ANY QUESTIONS?

...


I CODED IN PYTHON ONCE