python-2

September 15, 2020

0.1 # Introduction to Programming – Python

0.2 Course 2 – loops and iterables

0.2.1 ESSEC Business School

gael.guibon@gmail.com

gael.guibon@telecom-paris.fr

Original content inspired by Clement Plancq's IM courses

1 Summary

- **instructions** : `print()`
- **basic operators** : logic (`and or`), math (`+ = - /`), membership (`in not in`)
- **variables** : `name = city city = 'Cergy'`
- **types concepts and basics** and conversion : `type() str() bool() int() float()`
- **conditions and nested conditions** : indentations !

```
if cond:
    if cond:
        if cond:
            instruction
        elif:
            instruction
        else:
            instruction
    else:
        instruction
else:
    instruction
```

- Mini textual game

2 Lists

- Variable type
- Ordered structure
- Iterable
- Can contain multiple values
- Any types of values
- Values can be accessed by index (0 to n-1, n = number of elements)
- Index starts at 0

```
[ ]: # initialize (both methods aree equal)
students = [] # list initialization. Still empty
students = list() # another list initialization. Empty

# initialize with values
students = ['Brice','Ghita', 'Joshua', 'Maty', 'Quynh Chi', 'Julien'] #␣
 ↪initialize list with values (prefered way)
students = list(['Brice','Ghita', 'Joshua', 'Maty', 'Quynh Chi', 'Julien']) #␣
 ↪overkill !

# access
students[0] # access element by index
```

```
[ ]: # A list is a type
print( type( [ 'monday', 'tuesday', 'wednesday' ] ) )

# A list can have multiple types
superList = [ 'monday', 0, 23.5, 'hi!']
print(type(superList[0]), type(superList[1]), type(superList[2]) )
```

3 List indexes

Floors: 0. RDC (ground floor) 1. First floor 2. Second floor

Total floors : 3

4 List functions

- Lists have functions for basic operations fonctions propres
- Main functions :
  - `append(x)` : add an element at the end of the list
  - `extend([x, y, z])` : add all these elements at the end of the list
  - `pop([index])` : delete and return the last element of the list (or given indexes)
  - `remove(x)` : remove first element with value x
  - `index(x)` : return the index of the first element with value x

  - `count(x)` : return the number of occurrences for x
  - `sort()` : sort the list doc pour en savoir plus sur les ordres de tri

[ ]:
```
# init
students = ['Brice','Ghita', 'Joshua', 'Maty', 'Quynh Chi', 'Julien']
```

[ ]:
```
# append a student
students.append('Monica')
print(students)
```

[ ]:
```
# append multiple students
students.extend(['Enora', 'Ishita'])
print( students )
```

[ ]:
```
# delete and return the last element
mekhla = students.pop()
print(mekhla)
```

[ ]:
```
# remove first element with this value
students.remove('Monica')
print(students)
```

[ ]:
```
# sort list and get index
print('Brice is index', students.index('Brice'))
students.sort()
print(students)
print('Brice is now index', students.index('Brice'))
```

5 Length & sublists

- Length is obtained through built-in function `len()`
- Sublists can be obtained with index ranges `myList[:5]`. `:` mean 'to the edge' (start or end of the list)

[ ]:
```
# print the size of the list (i.e. number of students)
print( len( students ) )
```

[ ]:
```
# get first 5 elements
students[:5]
```

[ ]:
```
# get last 3 elements
students[3:]
```

[ ]:
```
# get elements between 2 and 5
someList = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print( someList[2:5])
```

6 Loops

- `while` loops need a condition to stop iterating.

```
while conditionIsTrue:
    instruction
```

- `for` loops can be applied to iterables types (`list()` for instance)

```
// classic loop in Java (and almost all programming languages)
for(int i = 0; i < listOfElements.length; i++){
    //instruction
}
```

```
# loop for each in Python
for element in listOfElements:
    instruction
```

[ ]:
```
# while loop
i = 0
while i < 5:
    print(i)
    i = i + 1  # need to change the value or else the condition will never be␣
 ↪true and the loop while never stop
```

[ ]:
```
# for loop
for n in range(5):
    print(n)
```

[ ]:
```
# for loop
for student in students: # will iterate ooever the students in order
    print(student)
```

7 For and While have different usages

- "print 'firstname:' before each student name"

[ ]:
```
for student in students:
    print('firstname:', student)
```

- "walk one step after another until you arrive at destination"

[ ]:
```
destination = 20
currentLocation = 14
while currentLocation < destination:
    print('walking...')
    currentLocation+=1 # increment. compact way to say currentLocation =␣
 ↪currentLocation + 1
```

```
print('Arrived at destination', destination)
```

- "while input is not one of the options continue asking"

```
[ ]: choices = ['adopt', 'kill', 'run']
choice = ""
while choice not in choices:
    choice = input()
    print("not recognized, please type one of the following options", choices)
    if(choice == 'quit'):
        break
    elif choice == 'adopt':
        print('You adopted the dog')
```

8 Skip or exit loops

- **continue**: in loops you can use `continue` to directly continue the loop (go to the next iteration).
- **break**: in loops you can use `break` to stop the loop and exit it.

```
[ ]: for student in ['Brice','Ghita', 'Joshua', 'Maty', 'Quynh Chi', 'Julien']:
    if student != 'Julien':
        print(student, 'is not Julien!')
        break # try it with 'continue' instead
    else:
        print('found him!')
```

9 Basic Role Playing Game (2)

- Objective: construct a textual role playing game. The game is only text with choices, conditions to verify the choices and player status.
- To do so you need one additional function (see below)

9.1 Rules

- Player advance from a room to another by textually selecting one of the rooms.
- Player starts with 200 hp (health points)
- A bad decision cost the player to lose 25 hp.
- **While** the player HP is greater than 0, continue playing
- **For** each actions, display it from a list of Strings

9.2 Input() Function

- Retrieve the player input

[ ]:
```
# Examples
# Get the input function and display a greeting message
print("Hello Player One, what's your name?")
playerName = input()
print("Welcome", playerName)

choice = ""
choices = ["first", "second"]
while choice not in choices:
    print('''Your are in a tiny room. Humidity fills the air but your stomach␣
↪reminds you that you are very hungry.
    You are in front of two doors. Behind the first one you can hear muffled␣
↪voices.
    Behind the second one you can smell something intriguing.''')
    print('Which door do you choose? Type', choices[0], 'for the first room␣
↪and', choices[1], 'for the second room.')
    choice = input()
    if choice == 'first':
        print('Game Over! Try again', playerName)
    elif choice == 'second':
        print('Upon opening the door, you can see a huge fest with exquisite␣
↪meals everywhere.')
    else:
        print('WRONG : Possible choices', choices)
```

- Complete the Role Playing Game skeleton by adding choices, player HP, etc.
- *Tip: this kind of game is a decision tree of choices.*
- Use while to keep the playing going on until certain conditions
- Use while to check the player hp
- Use for to display player' items or actions

10 Dictionaries

10.0.1 quick intro

- Dictionaries are another Type of variable `dict()`
- Follow only one concept: **Key and Value**
- Can contain deeper informations, with explicit hierarchy
- Data are not ordered (contrary to `list()`)
- Value can be accessed by its associated Key
- **Keys are unique** you cannot associate multiple values to one key.
- `keys()` returns the list of keys, `values()` returns the list of values

[ ]:
```python
# initialize an empty dictionary
dico = {}
dico = dict() # both ways are corrects

# intialize with values
dico = { "key" : "value" }
print(dico)
```

[ ]:
```python
# more concrete example
player = { "name":"gaël", "health":200 }
print(player)
```

11 Access data from a dict

- By specifying the key : number, boolean or String
- Commonly used with String keys

[ ]:
```python
player["name"]
```

[ ]:
```python
# get the list of keys
print( player.keys() )

# get the list of values
print( player.values() )
```

12 Insert or modify data in a dict

[ ]:
```python
# modify or create the key
player["items"] = ['bottle', 'laptop']
print(player)
```

[ ]:
```python
# delete a key : two methods
deletedValue = player.pop('items', None)
print(deletedValue)
print(player)

# if you are certain the key exists
player["items"] = ['bottle', 'laptop']
del player['items']
print(player)
```

13 Represent bigger data

- With variable types, especially dict() and list() you can handle more interesting data.
- You will often need to represent **nested data**

[ ]:
```
# example
user = {
    'name': 'john',
    'age': 19,
    'job': 'student',
    'friends': ['jack', 'paul'],
    'stats': {
        'social': 'friendly',
        'seriousness': 'bad',
        'formerEmployeee': False
    }
}
```

14 Install local python environnement

At home you may want to use Python locally. Here are the steps: 1. Download the latest Python (Python3 not 2) from here: `https://www.python.org/downloads/` 2. Install it (if you are using Ubuntu 16.04 you already have Python installed)

Start using it: - Open terminal(unix/mac) or CommandLine(windows), type `python` to start an interactive python environnement - Create a file named `my_super_program.py` and type `print('hello')` inside. Execute this file by typing `python3 my_super_program.py`.

To code you may need an IDE for smoother coding. I would suggest Visual Studio Code. For a python only IDE the best one would be PyCharm.