python-3

September 15, 2020

0.1 # Introduction to Programming – Python

0.2 Course 3 – dictionaries & functions

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 !
- **lists and lists functions** : `list()`, `myList = ['hello', 'world']`, `myList.append()`, `len(myList)` etc.
- **loops** : while `while conditionIsTrue:` and for `for element in elementList:`, `break`, `continue`
- Mini textual game

2 Dictionaries

- 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 (you just need to change its type to list())

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

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

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

student = {'name':'zhanyi', 'age'.... 'friends':['friend1', 'friend2'] }
```

3 Access data from a dict

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

[ ]:
```
player["name"] = 'whatever'
print(player['name'])
print(player)
```

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

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

4 Insert or modify data in a dict

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

[ ]:
```
# 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)
```

5 Explore a dict()

[ ]:
```
# first populate the dict
# player = {'name':'gael',   'inventory': ['rose', 'pencil', 'smartphone'],  ␣
 ↪'grade':'noob'}
player = dict()
player['name'] = 'gael'
player['inventory'] = ['rose', 'pencil', 'smartphone']
player['grade'] = 'noob'
print('player =', player)
# get list of keys
print( player.keys(), type(player.keys()), len(player.keys()) )
# get list of values
print( player.values(), type(player.values()), len(player.values()) )
```

5.0.1 Check if a key exist:

[ ]:
```
nameFieldExists = 'name' in player
print( 'Does the "name" field exists in the dictionary "player" ?',␣
 ↪nameFieldExists)
```

[ ]:
```
itemsFieldExists = 'items' in player
print( 'Does the "items" field exists in the dictionary "player" ?',␣
 ↪itemsFieldExists)
```

5.0.2 Iterate over fields in a dict

[ ]:
```
for key, value in player.items():
    print('key =', key, ' ; value =', value)
```

5.0.3 Get or add field and value

`setdefault(key, defaultValue)` returns the value associated to the key. If the key does not exist, it adds the key associated to `defaultValue`.

[ ]:
```
player.setdefault('age', 20)
print(player)
```

5.0.4 Safely access non present field

Using `player[key]` will yield an error. You can safely try to access a field by using `player.get(key)`. Returns the value if the key exists, else returns None.

`(in pseudo algo) IF key exists: RETURN the associated value ELSE: RETURN None`

[ ]:
```
print( player['unknownKey'] ) # gives an error
```

[ ]:
```
print(player)
print( player.get('unknownKey')) # returns None
print( player.get('name') ) # it exists
```

6 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', # str()
    'age': 19, # int()
    'money': 24.56, # float()
    'job': 'student', # str()
    'friends': ['jack', 'paul'], # list() of two str()
    'stats': {
        'social': 'friendly', # str()
        'seriousness': 'bad', # str()
        'formerEmployeee': False # bool()
    } # dict()
} # dict()

print(user)
```

7 Basic Role Playing Game (3)

- Objective: construct a textual role playing game. The game is only text with choices, conditions to verify the choices and player status.

7.1 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.
- For each actions, display it from a list of Strings
- **Represent data as dictionaries or a big nested dict()** (see example below)

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

choice = ''
events = {
    'first': {
        'message': 'Game Over! Try again' + player['name']
    },
    'second': {
        'messages': ['Upon opening the door, you can see a huge fest with␣
 ↪exquisite meals everywhere.', 'heyy'],
        'follow':{}
    }
}
pastEvents = []

while choice not in events.keys():
    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', list(events.keys())[0], 'for the␣
 ↪first room and', list(events.keys())[1],
          'for the second room.')
    choice = input()
    if choice in events.keys():
        pastEvents.append(choice)
        if type(events[choice]['message']) == str():
        if 'messages' in events[choice]:
            for message in events[choice]['messages']:
                print(message)
        else:
            print(events[choice]['message'])
    else:
        print('WRONG : Possible choices', list(events.keys()) )

print(pastEvents)
```

- Complete the Role Playing Game skeleton by adding choices, player HP, etc.
- *Tip: this kind of game is a decision tree of choices.* : the decision tree is represented as a

nested dictionary
- Use while to keep the playing going on until certain conditions
- Use for to display player' items or actions

8 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.

9 Functions

9.0.1 Quick intro

A function is a subprogram, a way to reuse the same code again and again by just calling it.

Functions have: - A name - Arguments (if specified)

Functions do: - A process - Return something (if specified)

Use *docstrings* to documentate functions:

`""" this function does this process and return that value but need those arguments """`

[ ]:
```
def mySuperbFunction(arg1, arg2):
    """
    This superb function needs 2 arguments.
    It returns None (such a shame! this function is still useless)
    """
    return None
```

Examples This function return the sum of the three arguments.

[ ]:
```
def sumItUp(arg1, arg2, arg3):
    """
    Well, this is but a sum of the 3 args.
    """
    return arg1 + arg2 + arg3
```

[ ]:
```
# Now that I defined my function I can use it like this
resultSum = sumItUp(2,3,-5)
print(resultSum)
```

9.1 Functions optional arguments with default value

- Some arguments can have a default value, hence their declaration is optional.
- Mandatory args always comes before optional ones

[ ]:
```
# DEFine the function
def createStudent( age, grade, name='noname'):
    """ This creates a student dict with default name as 'noname' """
    return {'name':name, 'age':age, 'grade':grade}

# now use it
print( createStudent(19, 20) )
print( createStudent( 19, 20, name='superStudent' ) )
# as you can see, the name arg is optional
```

[ ]:
```
# this will not work due to the optional arg placed before a mandatory one
createStudent(name='thisStudent', 19, 20)
```

[ ]:
```
# this will not work due to the lack of mandatory args
createStudent(19)
```

10 Basic Role Playing Game (4)

- Objective: construct a textual role playing game. The game is only text with choices, conditions to verify the choices and player status.

10.1 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.
- For each actions, display it from a list of Strings
- **Represent data as dictionaries or a big nested dict()** (see example below)
- **Use functions** to remove code duplicates and reduce the line numbers

[ ]:
```
# Example with dict and basic functions
player = dict()
print("Hello Player One, what's your name?")
player['name'] = input()
print("Welcome", player['name'])
```

```
events = {
    'first': {
        'message': 'Game Over! Try again ' + player['name']
    },
    'second': {
        'message': '''Upon opening the door, you can see a huge fest with␣
 ↪exquisite meals everywhere.
                    Will you eat it?''',
        'paths': {
            'yes': { 'message': 'You died from poison'},
            'no': { 'message': '`You died from hunger'}
        }}
}
pastChoices = []



def askAction(event, choices):
    '''
    Ask an action from the user. Stay alive while the action is not recognized.
    '''
    choice = ''
    while choice not in choices:
        choice = input()
        if choice in choices:
            print(event[choice]['message'])
            pastChoices.append(choice)
            if('paths' in event[choice]):
                askAction(event[choice]['paths'], list(event[choice]['paths'].
 ↪keys()) )
        else:
            print('WRONG : Possible choices', choices )
    return True

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', list(events.keys())[0], 'for the first␣
 ↪room and', list(events.keys())[1],
              'for the second room.')

askAction(events, list(events.keys()) )

print('You finished playing the game. Here are your past actions :', pastChoices)
```