

# Sprint 01

Bruno Augusto, Iago Lincon, João Gabriel, Maycow dos Santos

25/04/2017

## **Resumo**

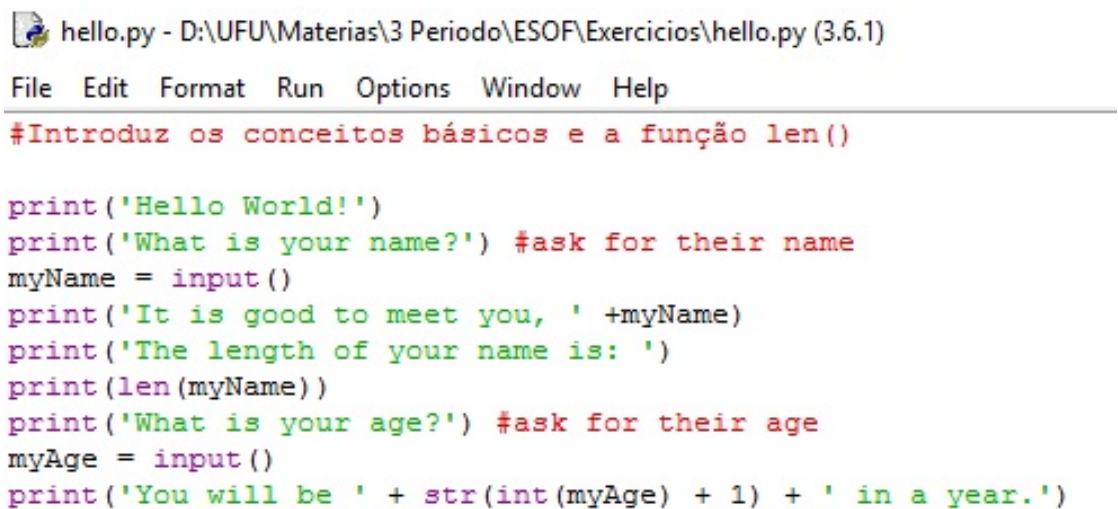
Foi designado ao grupo aprender o básico da programação em Python nos seis primeiros capítulos iniciais do livro originalmente nomeado como "Automate the Boring Stuff". Em tais capítulos, é ensinado desde comandos simples como a concatenação proporcionado pelo Python, até a existência de funções, listas, dicionários e suas funcionalidades. Posteriormente, o grupo deveria escolher dois capítulos da parte destinada a programação funcional do livro e aprendê-los para começar a adquirir o conhecimento do que a linguagem proporcionada pelo software trabalhado pode fazer.

# Parte I:

## Adicionando Conhecimento

# 1 O básico do Python

O capítulo introduz as expressões matemáticas que são utilizadas dentro da programação do software, noção das variáveis inteiras, pontos flutuantes e strings, concatenação entre strings e o conhecimento da introdução de valores para variáveis.



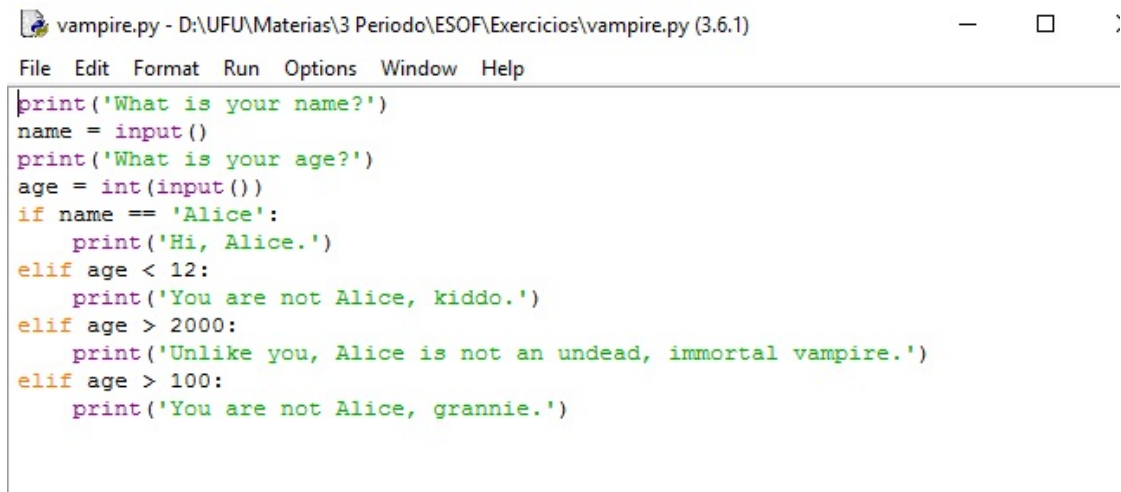
```
hello.py - D:\UFU\Materias\3 Periodo\ESOF\Exercicios\hello.py (3.6.1)
File Edit Format Run Options Window Help
#Introduz os conceitos básicos e a função len()

print('Hello World!')
print('What is your name?') #ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is: ')
print(len(myName))
print('What is your age?') #ask for their age
myAge = input()
print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

Figura 1: Programa que calcula o tamanho do nome do usuário e diz sua idade no ano seguinte

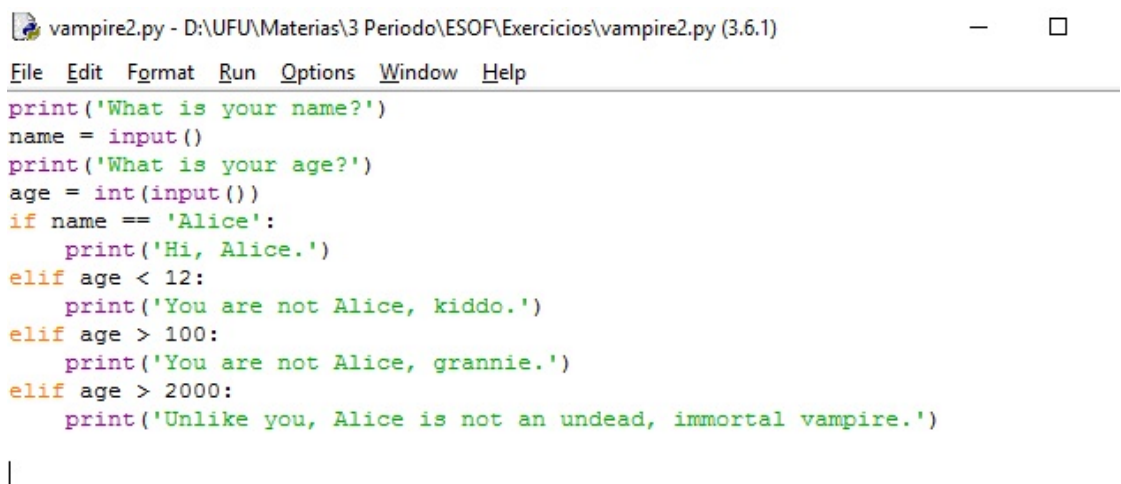
# 2 Controle de Fluxo

Agora que o mais básico da programação já fora introduzido, é hora de aprender sobre os booleanos, condições para controle de fluxo de dados ou loops, a função range() e o conhecimento sobre a importação de funções do Python, como o *import sys* (para o *sys.exit()*).



```
vampire.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\vampire.py (3.6.1)
File Edit Format Run Options Window Help
print('What is your name?')
name = input()
print('What is your age?')
age = int(input())
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
elif age > 2000:
    print('Unlike you, Alice is not an undead, immortal vampire.')
elif age > 100:
    print('You are not Alice, grannie.')
```

Figura 2: Baseado em condições, o programa diz se o usuário é Alice ou não



```
vampire2.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\vampire2.py (3.6.1)
File Edit Format Run Options Window Help
print('What is your name?')
name = input()
print('What is your age?')
age = int(input())
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
elif age > 100:
    print('You are not Alice, grannie.')
elif age > 2000:
    print('Unlike you, Alice is not an undead, immortal vampire.')
```

Figura 3: O programa *vampire v2.0* corrige o erro da condição  $\text{age} \geq 100$  e  $\text{age} \geq 2000$

exitexample.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\exitexample.py (3.6.1)

File Edit Format Run Options Window Help

```
import sys
while True:
    print('Type exit to exit.')
    response = input()
    if response == 'exit':
        sys.exit()
    print('You typed ' + response + '.')
```

Figura 4: Demonstrando a função *sys.exit()*

### 3 Funções

Uma parte muito importante da organização de programas (tanto esteticamente, como funcionalmente) está imposto neste capítulo. A introdução do conceito de funções e seus objetivos.

A partir de agora, os programas começam a ficar um pouco mais interessantes e com multifuncionalidades que já foram aprendidas em capítulos anteriores.

sameName.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\sameName.py (3.6.1)

File Edit Format Run Options Window Help

```
def spam():
    eggs = 'spam local'
    print(eggs)

def bacon():
    eggs = 'bacon local'
    print(eggs)
    spam()
    print(eggs)

eggs = 'global'
bacon()
print(eggs)
```

Figura 5: Introdução ao conceito de funções

sameName2.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\sameName2.py (3.6.1)

File Edit Format Run Options Window Help

```
def spam():
    global eggs
    eggs = 'spam'

eggs = 'global'
spam()
print(eggs)
```

Figura 6: Global Scope introduzido no programa anterior

guessTheNumber.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\guessTheNumber.py (3.6....

File Edit Format Run Options Window Help

```
#Programa de "descobrir o número"
import random #funcao python para gerar valores aleatorios
secretNumber = random.randint(1,20) #a variavel ira receber um valor aleatorio
print('I am thinking of a number between 1 and 20.')

#A interface pergunta para o usuário dar um palpite 6 vezes
for guessesTaken in range(1,7):
    print('Take a guess.')
    guess = int(input())

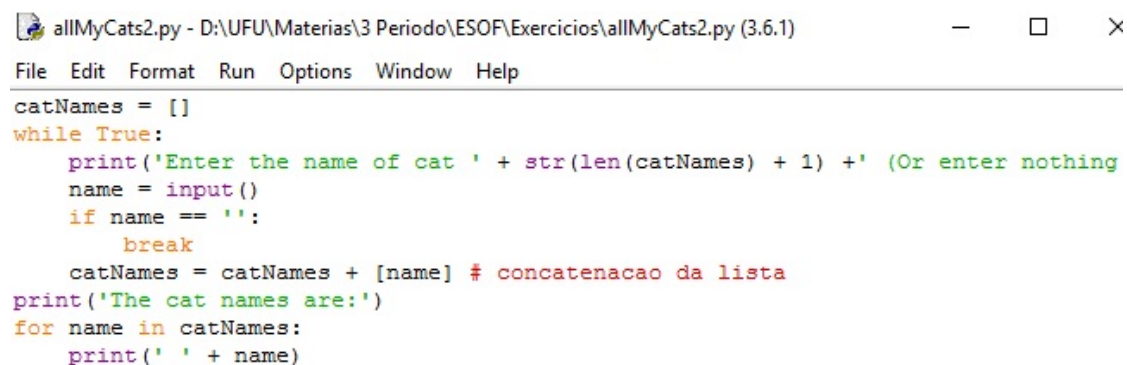
    if guess < secretNumber:
        print('Your guess is too low.')
    elif guess > secretNumber:
        print('Your guess is too high.')
    else:
        break #o loop encerra caso o usuario acerte o valor

if guess == secretNumber:
    print('Good job! You guessed my number in ' + str(guessesTaken) + ' guesses!')
else:
    print('Nope. The number I was thinking of was ' + str(secretNumber))
```

Figura 7: Programa que reúne todos conceitos estudados até o momento

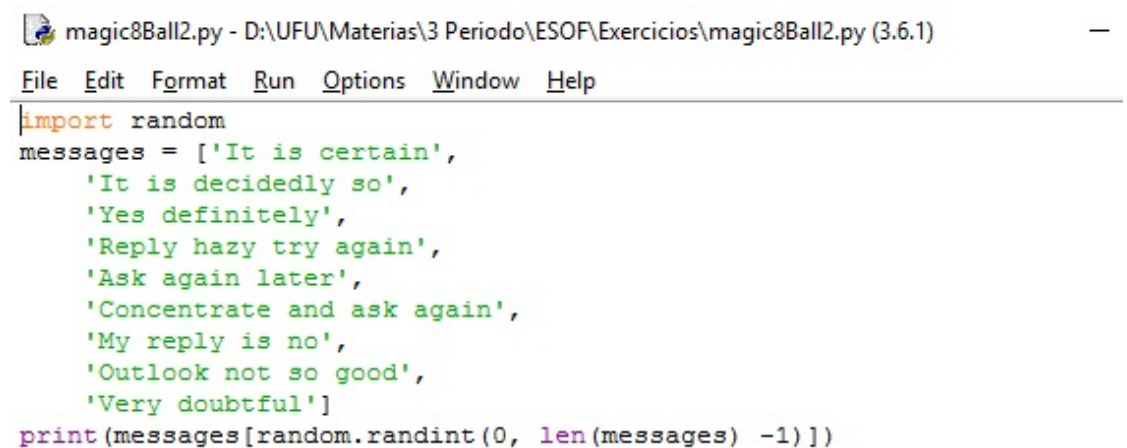
## 4 Listas

Muitas vezes não queremos trabalhar com uma infinidade de variáveis, atribuindo um único valor para cada. Na verdade, na maioria das vezes, precisamos de uma variável que suporta vários valores. No Python, essa função é apresentada pelas listas e tuplas.



```
allMyCats2.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\allMyCats2.py (3.6.1)
File Edit Format Run Options Window Help
catNames = []
while True:
    print('Enter the name of cat ' + str(len(catNames) + 1) + ' (Or enter nothing)')
    name = input()
    if name == '':
        break
    catNames = catNames + [name] # concatenacao da lista
print('The cat names are:')
for name in catNames:
    print(' ' + name)
```

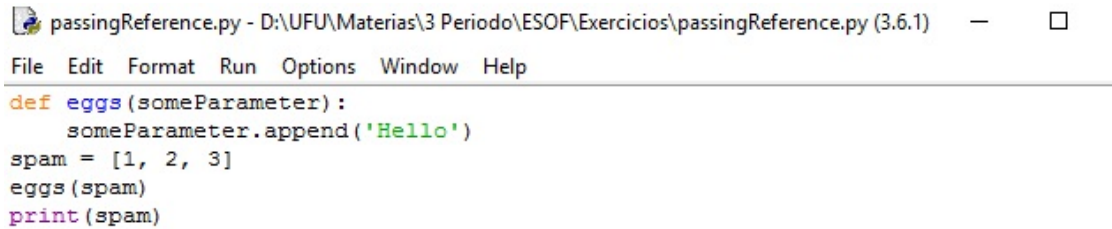
Figura 8: Programa usado para concatenar os nomes dos bichanos em uma lista



```
magic8Ball2.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\magic8Ball2.py (3.6.1)
File Edit Format Run Options Window Help
import random
messages = ['It is certain',
            'It is decidedly so',
            'Yes definitely',
            'Reply hazy try again',
            'Ask again later',
            'Concentrate and ask again',
            'My reply is no',
            'Outlook not so good',
            'Very doubtful']
print(messages[random.randint(0, len(messages) - 1)])
```

Figura 9: Função `emphrandom` para produzir um jogo de aleatoriedade





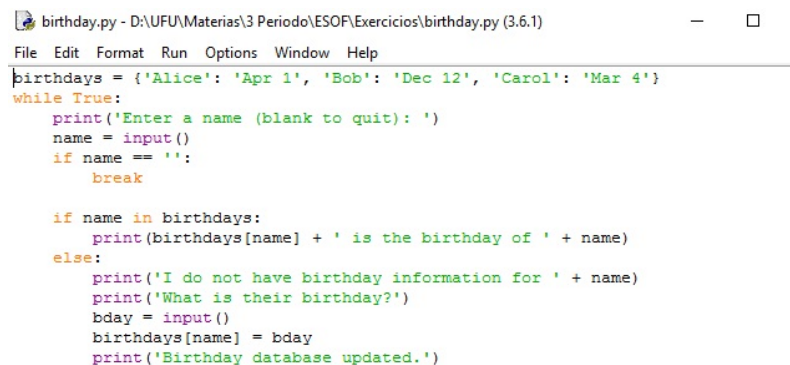
```
passingReference.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\passingReference.py (3.6.1)
File Edit Format Run Options Window Help
def eggs(someParameter):
    someParameter.append('Hello')
spam = [1, 2, 3]
eggs(spam)
print(spam)
```

Figura 10: Demonstração da transferência de referência através de variáveis locais em uma função

## 5 Estruturando Dados e Dicionários

Dicionários são extensões de listas, com algumas vantagens e desvantagens dependendo da necessidade do programador. Em geral, eles se tornam muito úteis, principalmente pelo esquema "chave-valor" que é atribuído a cada dicionário.

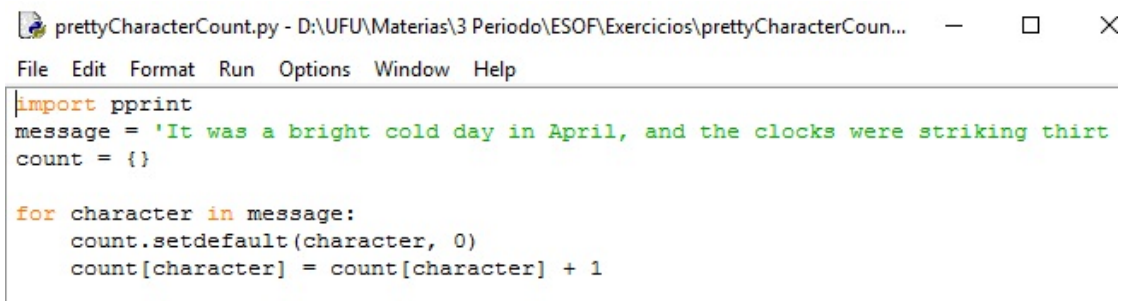
Além disso, possuem a possibilidade de construção dic.-dic. (um dicionário dentro de outro dicionário), função que pode ser muito útil quando você precisa atribuir valores para duas "chaves" diferentes. Esse processo pode simplificar muito a programação de softwares complexos pela acessibilidade que você pode trabalhar com os seus itens.



```
birthday.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\birthday.py (3.6.1)
File Edit Format Run Options Window Help
birthdays = {'Alice': 'Apr 1', 'Bob': 'Dec 12', 'Carol': 'Mar 4'}
while True:
    print('Enter a name (blank to quit): ')
    name = input()
    if name == '':
        break

    if name in birthdays:
        print(birthdays[name] + ' is the birthday of ' + name)
    else:
        print('I do not have birthday information for ' + name)
        print('What is their birthday?')
        bday = input()
        birthdays[name] = bday
        print('Birthday database updated.')
```

Figura 11: Usando um dicionário, criamos uma função que mostra os aniversários dos amigos. Caso este não esteja na lista, ele irá adicioná-lo no dicionário

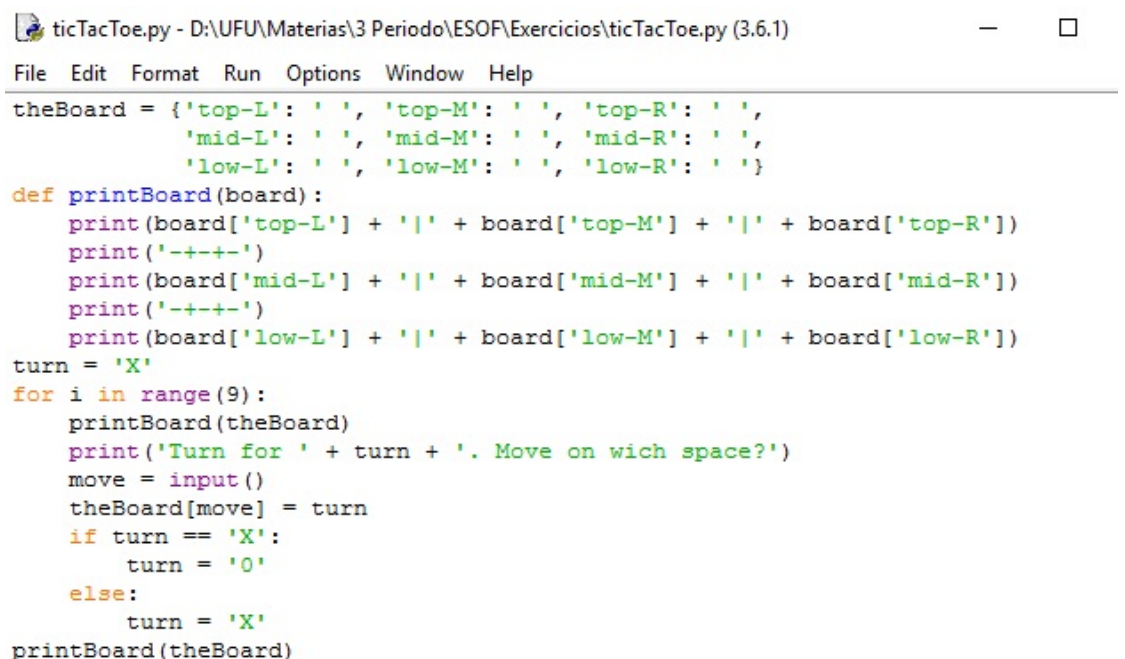


```
prettyCharacterCount.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\prettyCharacterCoun...
File Edit Format Run Options Window Help

import pprint
message = 'It was a bright cold day in April, and the clocks were striking thirt
count = {}

for character in message:
    count.setdefault(character, 0)
    count[character] = count[character] + 1
```

Figura 12: for usado juntamente com a função *pprint* para contar caracteres em uma variável de string



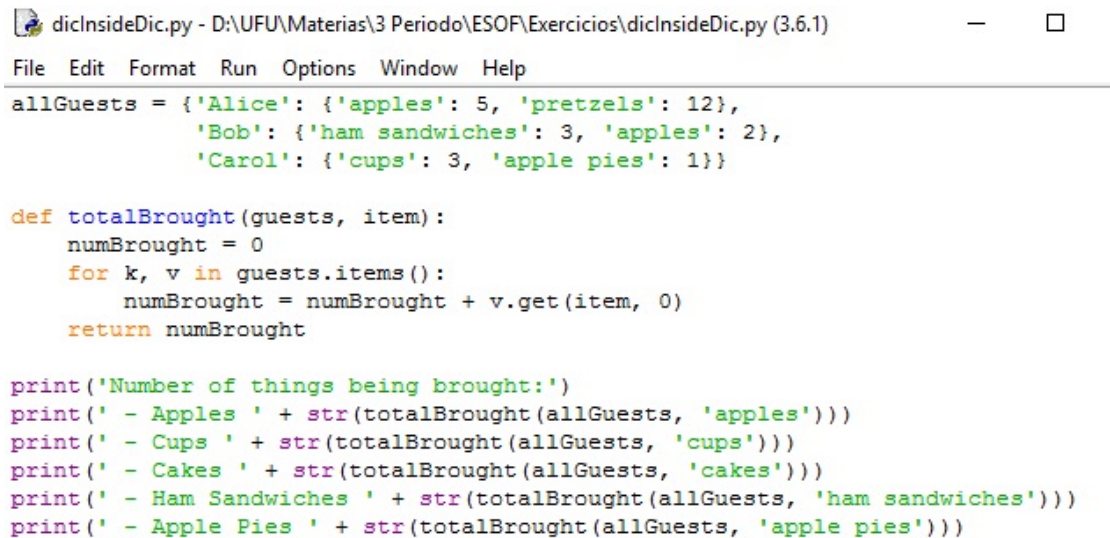
```
ticTacToe.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\ticTacToe.py (3.6.1)
File Edit Format Run Options Window Help

theBoard = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
            'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
            'low-L': ' ', 'low-M': ' ', 'low-R': ' '}

def printBoard(board):
    print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
    print('-+-+-')
    print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
    print('-+-+-')
    print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])

turn = 'X'
for i in range(9):
    printBoard(theBoard)
    print('Turn for ' + turn + '. Move on wich space?')
    move = input()
    theBoard[move] = turn
    if turn == 'X':
        turn = 'O'
    else:
        turn = 'X'
printBoard(theBoard)
```

Figura 13: Aplicando os conhecimentos adquiridos para desenvolver um protótipo de jogo da velha



```

dicInsideDic.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\dicInsideDic.py (3.6.1)
File Edit Format Run Options Window Help
allGuests = {'Alice': {'apples': 5, 'pretzels': 12},
             'Bob': {'ham sandwiches': 3, 'apples': 2},
             'Carol': {'cups': 3, 'apple pies': 1}}

def totalBrought(guests, item):
    numBrought = 0
    for k, v in guests.items():
        numBrought = numBrought + v.get(item, 0)
    return numBrought

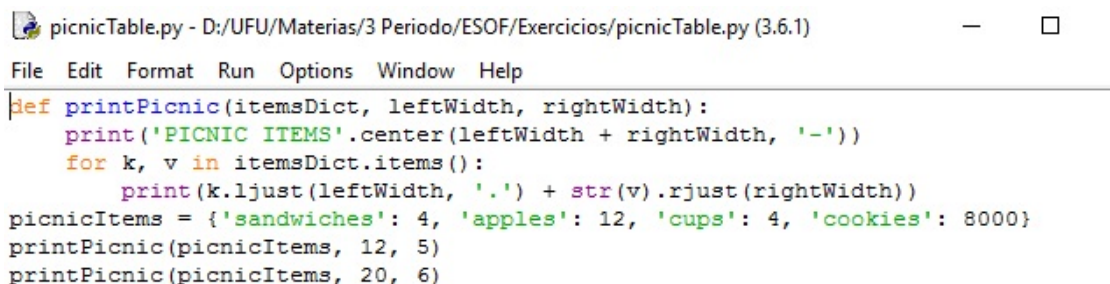
print('Number of things being brought:')
print(' - Apples ' + str(totalBrought(allGuests, 'apples')))
print(' - Cups ' + str(totalBrought(allGuests, 'cups')))
print(' - Cakes ' + str(totalBrought(allGuests, 'cakes')))
print(' - Ham Sandwiches ' + str(totalBrought(allGuests, 'ham sandwiches')))
print(' - Apple Pies ' + str(totalBrought(allGuests, 'apple pies')))

```

Figura 14: Programa para demonstrar um dicionário dentro de outro dicionário. Nele ele diz a quantidade de cada item que fora levado para o picnic, caso esse não exista, então = 0

## 6 Manipulando Strings

Esta seção é destinada para o aprofundamento do seu embasamento teórico sobre as funcionalidades do Python. São introduzidas várias funções que trabalham para modelar sua string (e seus processos) da forma qual for mais conveniente para seu projeto.

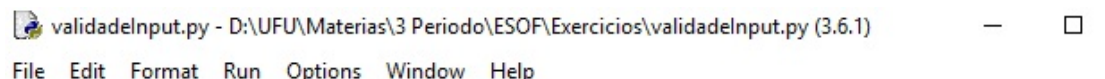


```

picnicTable.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\picnicTable.py (3.6.1)
File Edit Format Run Options Window Help
def printPicnic(itemsDict, leftWidth, rightWidth):
    print('PICNIC ITEMS'.center(leftWidth + rightWidth, '-'))
    for k, v in itemsDict.items():
        print(k.ljust(leftWidth, '.') + str(v).rjust(rightWidth))
picnicItems = {'sandwiches': 4, 'apples': 12, 'cups': 4, 'cookies': 8000}
printPicnic(picnicItems, 12, 5)
printPicnic(picnicItems, 20, 6)

```

Figura 15: Faz uma tabela com todos os itens do picnic utilizando as funções `.center()`, `.ljust()`, `.rjust()`



```
validadeInput.py - D:\UFU\Materias\3 Período\ESOF\Exercícios\validadeInput.py (3.6.1)
File Edit Format Run Options Window Help
while True:
    print('Enter your age: ' )
    age = input()
    if age.isdecimal():
        break
    print('Please enter a number for your age.')

while True:
    print('Select a new password (letters and numbers only):' )
    password = input()
    if password.isalnum():
        break
    print('Passwords can only have letters and number.')
```

Figura 16: Demonstração da função *.isdecimal()*

## Parte II:

## Primeiras Tarefas

## 7 Correspondência de Padrões com as Expressões Regulares

Introduz o conhecimento de uma poderosa ferramenta do Python, a Regular Expressions, ou re (import re). Processos que envolveriam diversas iterações e condições, agora podem ser feitos com uma pequena linha de código como a `.search()` ou `re.compile(r'.....')`.

isPhoneNumber.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\isPhoneNumber.py (3.6.1)

File Edit Format Run Options Window Help

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False
    for i in range(0, 3):
        if not text[i].isdecimal():
            return False
    if text[3] != '-':
        return False
    for i in range(4, 12):
        if not text[i].isdecimal():
            return False
    return True

print('012986584722 is a phone number?')
print(isPhoneNumber('012986584722'))
print('Roorona Zero is a phone number?')
print(isPhoneNumber('Roorona Zero'))
```

(a) isPhoneNumber v.01

isPhoneNumber2.py - D:\UFU\Materias\3 Período\ESOF\Exercicios\isPhoneNumber2.py (3.6.1) — □

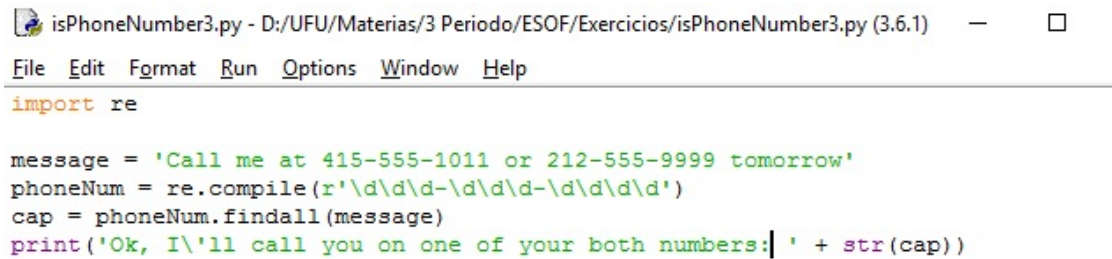
File Edit Format Run Options Window Help

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False
    for i in range(0, 3):
        if not text[i].isdecimal():
            return False
    if text[3] != '-':
        return False
    for i in range(4, 7):
        if not text[i].isdecimal():
            return False
    if text[7] != '-':
        return False
    for i in range(8, 12):
        if not text[i].isdecimal():
            return False
    return True

message = 'Call me at 415-555-1011 tomorrow. 415-555-9999 is my office.'
for i in range(len(message)):
    chunk = message[i:i+12]
    if isPhoneNumber(chunk):
        print('Phone number found: ' + chunk)
print('Done')
```

(b) isPhoneNumber v.02

Figura 17: Implementação da função que verifica números telefônicos com uma iteração que capta os números automaticamente



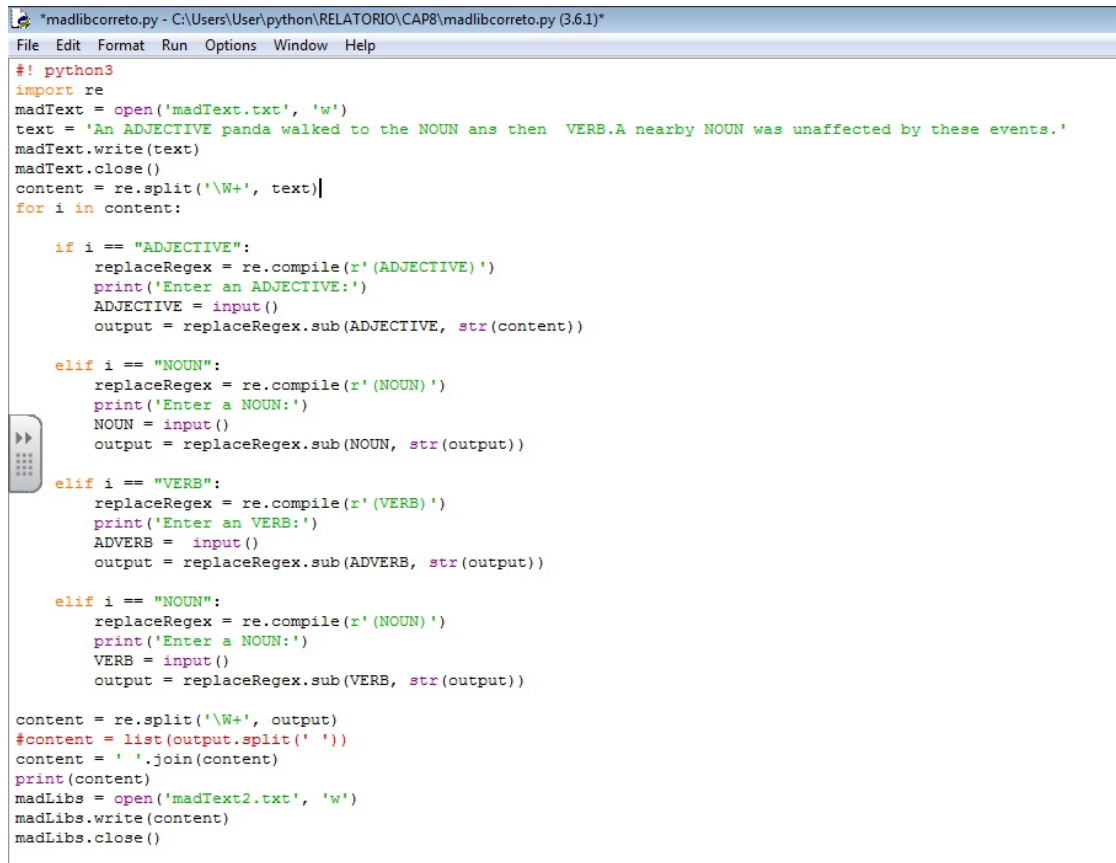
```
isPhoneNumber3.py - D:/UFU/Materias/3 Período/ESOF/Exercicios/isPhoneNumber3.py (3.6.1)
File Edit Format Run Options Window Help
import re

message = 'Call me at 415-555-1011 or 212-555-9999 tomorrow'
phoneNumber = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
cap = phoneNumber.findall(message)
print('Ok, I\'ll call you on one of your both numbers:|' + str(cap))
```

Figura 18: As duas versões dos programas anteriores simplificadas agora com a extensão *re*

## 8 Lendo e Escrevendo Arquivos

Neste capítulo o foco principal é aprender a interagir com o seu software de sistema operacional. Com este conteúdo, ficará fácil manipular arquivos desejados no seu OS através de algumas linhas de código do seu Python.



```
#!/usr/bin/python3
import re
madText = open('madText.txt', 'w')
text = 'An ADJECTIVE panda walked to the NOUN and then VERB. A nearby NOUN was unaffected by these events.'
madText.write(text)
madText.close()
content = re.split('\W+', text)
for i in content:

    if i == "ADJECTIVE":
        replaceRegex = re.compile(r'(ADJECTIVE)')
        print('Enter an ADJECTIVE:')
        ADJECTIVE = input()
        output = replaceRegex.sub(ADJECTIVE, str(content))

    elif i == "NOUN":
        replaceRegex = re.compile(r'(NOUN)')
        print('Enter a NOUN:')
        NOUN = input()
        output = replaceRegex.sub(NOUN, str(output))

    elif i == "VERB":
        replaceRegex = re.compile(r'(VERB)')
        print('Enter an VERB:')
        ADVERB = input()
        output = replaceRegex.sub(ADVERB, str(output))

    elif i == "NOUN":
        replaceRegex = re.compile(r'(NOUN)')
        print('Enter a NOUN:')
        VERB = input()
        output = replaceRegex.sub(VERB, str(output))

content = re.split('\W+', output)
#content = list(output.split(' '))
content = ' '.join(content)
print(content)
madLibs = open('madText2.txt', 'w')
madLibs.write(content)
madLibs.close()
```

Figura 19: Com todo o conhecimento dos caps. anteriores, o programa recebe substantivos, advérbios ou verbos e junta os elementos digitados pelo usuário formando uma frase ou um texto qual o usuário deseja