# Python in 4 sittings: Sitting 2

## Introduction to Python

### Summary of the second class

20th June, 2021



feel free to contact us;

arabindo@protonmail.com

kaustavbasu97@gmail.com

# Part A

# Ternary Operator

At first we will revisit simple `if-else` statement and start optimizing it, which in turn lead us to Ternary Operator

In [1]:
```
# So let's starrt with simple odd even program

number = 7

if(number%2==0):
    print('Even')
else:
    print('Odd')
```

```
Odd
```

In [2]:
```
# Functional Approach

def oddEvenCheck(num):
    num = int(num)
    if (num % 2 == 0):
        print('Even')
    else:
        print('Odd')

takeInput = int(input("Enter number: "))
oddEvenCheck(takeInput)
```

```
Enter number: 10
Even
```

In [3]:
```
# You can do it with fewer line
# When executing a function,
# after it reaches the return statement,
# Control is passed to the calling function
```

```
def oddEvenCheck1(num):
    num = int(num) # conv to num
    if (num % 2 == 0):
        return 'Even'
    return 'Odd'

takeInput = int(input("Enter number: "))
oddEvenCheck(takeInput)
```

```
Enter number: 15
Odd
```

In [4]:

```
# sum up in one line
# Tha's your ternary operation :)

def oddEvenCheckT(num):
    num = int(num)

    """
    --- Generic If-Else Syntax ---
    If(Condition is True):
        :return: 1st Option
    Else:
        :return: 2nd Option

    --- Ternary Op Syn ---
    1stOption if CondTrue Else 2ndOption
    """

    return 'Even' if num%2==0 else 'Odd'

print(oddEvenCheckT(input('num: ')))
```

```
num: 25
Odd
```

# RegEx is here with out much explanation. We will provide a better note later.

You can look into the official documentation: https://docs.python.org/3/library/re.html

In [5]:

```
"""
Create a indian mobile number validator.

Constraints/Condn/RQs:
1. 1st digit should not be anything other than 0 or +91
    or none followed by on of {9,8,7,6,5}
2. Exactly 10 digits should be there.
"""

import re

test_string = input('Enter your number: ')

pattern1 = '^(0[5-9]|\+91[5-9]|91[5-9])[0-9][0-9][0-9][0-9]' \
           '[0-9][0-9][0-9][0-9][0-9]$'
result1 = re.match(pattern1, test_string)
print('1: passed' if result1 else '1: failed')

pattern2 = '.........'
```

```
result2 = re.match(pattern2, test_string)
print('2: passed' if result2 else '2: failed')
```

```
Enter your number: 9165897458
1: failed
2: passed
```

# Part B

# Application of Dictionary

## Design a Tic-toc-toe game

The idea is to have a function which will print the tic toc toe board using the dictionary values. The dictionary value will be updated based on user input.

On every iteration user will iput the the position of his turn, like 00, 01 and so on.

Position table(Alternatively, you can name them as 11,12,13,21,22,23,31,32,33...It's completely your choice):

| | col1 | col2 | col3 |
|---|---|---|---|
| row1 | 00 | 01 | 02 |
| row2 | 10 | 11 | 12 |
| row3 | 20 | 21 | 22 |

We will print the board in the following way:

```
   | |
  -+-+-
   | |
  -+-+-
   | |
```

In [1]:

```
# Let's Define the printBoard function first
# This will help us to print the board from dictionary value
# We will pass a dictionary as an argumet

def printBoard(board):
    print(board['00']+'|'+board['01']+'|'+board['02'])
    print('-'+'+'+'-'+'+'+'-')
    print(board['10']+'|'+board['11']+'|'+board['12'])
    print('-'+'+'+'-'+'+'+'-')
    print(board['20']+'|'+board['21']+'|'+board['22'])

# Now let's create a dictionary, which will hold the values
tic = {'00': ' ', '01': ' ', '02':' ',
        '10': ' ', '11': ' ', '12':' ',
        '20': ' ', '21': ' ', '22':' '}
printBoard(tic) # Print the board
```

```
 | |
-+-+-
 | |
```

```
-+-+-
 | |
```

In [2]:

```
# Implement the game
# This implementation have some short comings,
# We'll refer those in the problem sheet.
turn = 'x'
for i in range(9):
    print("turn no:", i)
    if (turn == 'x'):
        pos = input("It's x turn, enter the position: ")
        # check whether theposition is already taken
        if (tic[pos]==" "):
            tic[pos] = 'x'
            turn = 'o'
        else:
            print('wrong choice')
    else :
        pos = input("It's o turn, enter the position: ")
        if (tic[pos]==" "):
            tic[pos] = 'o'
            turn = 'x'
        else:
            print('wrong choice')
    printBoard(tic)
```

```
turn no: 0
It's x turn, enter the position: 11
 | |
-+-+-
 |x|
-+-+-
 | |
turn no: 1
It's o turn, enter the position: 12
 | |
-+-+-
 |x|o
-+-+-
 | |
turn no: 2
It's x turn, enter the position: 20
 | |
-+-+-
 |x|o
-+-+-
x| |
turn no: 3
It's o turn, enter the position: 22
 | |
-+-+-
 |x|o
-+-+-
x| |o
turn no: 4
It's x turn, enter the position: 00
x| |
-+-+-
 |x|o
-+-+-
x| |o
turn no: 5
It's o turn, enter the position: 02
x| |o
-+-+-
 |x|o
-+-+-
```

```
x| |o
turn no: 6
It's x turn, enter the position: 01
x|x|o
-+-+-
 |x|o
-+-+-
x| |o
turn no: 7
It's o turn, enter the position: 10
x|x|o
-+-+-
o|x|o
-+-+-
x| |o
turn no: 8
It's x turn, enter the position: 21
x|x|o
-+-+-
o|x|o
-+-+-
x|x|o
```

# Counting character and words from a string

## Before that, we will need some tool

In [3]:
```
# Let's get back to our old game
game = {'gem': 20, 'life': 3, 'user':'adghkj'}
print("before: ", game)
"""
There is no key called 'token'
We will check if there is a key in the dictionary
If not we will assign some value to it
 * Instead of # we can use three pairs of double inverted comma for
 multiline comments.
"""
if 'token' not in game:
    game['token'] = 12
print("after:", game)
```

```
before:  {'gem': 20, 'life': 3, 'user': 'adghkj'}
after: {'gem': 20, 'life': 3, 'user': 'adghkj', 'token': 12}
```

In [4]:
```
# We can do the same job with .setdefault() method
# It took two argument,
# first the key value to check
# second the default value to set
game.setdefault('number', 5)
game.setdefault('token', 0)
print('notice the token key', game)
# It haven't changed, because it is already in the dictionary
```

```
notice the token key {'gem': 20, 'life': 3, 'user': 'adghkj', 'token': 12, 'n
umber': 5}
```

## Let's get back to the business

Now let's say we have the following string in our `message` variable

```
message="JupyterLab is a web-based interactive development
environment for Jupyter notebooks, code, and data. JupyterLab is
flexible: configure and arrange the user interface to support a wide
range of workflows in data science, scientific computing, and machine
learning. JupyterLab is extensible and modular: write plugins that
add new components and integrate with existing ones."
```

We want to count the repeatation of the character in our string

In [5]:
```
message = "JupyterLab is a web-based interactive development environment for

count = dict() #It'll create a empty dictionary
for char in message:
    # If the character is not in the dictionary, add it!
    count.setdefault(char, 0) # set the repeation to 0
    count[char] += 1 # Increase it by 1, every time it reoccurs
print(count)
```

```
{'J': 4, 'u': 10, 'p': 10, 'y': 4, 't': 26, 'e': 40, 'r': 19, 'L': 3, 'a': 2
6, 'b': 8, ' ': 52, 'i': 25, 's': 15, 'w': 7, '-': 1, 'd': 14, 'n': 30, 'c':
11, 'v': 3, 'l': 8, 'o': 18, 'm': 6, 'f': 7, 'k': 2, ',': 4, '.': 3, 'x': 3,
':': 2, 'g': 8, 'h': 4}
```

In [6]:
```
# We can print it in a a better way,
#  for that we will need pretty printing library
import pprint
pprint.pprint(count)
```

```
{' ': 52,
 ',': 4,
 '-': 1,
 '.': 3,
 ':': 2,
 'J': 4,
 'L': 3,
 'a': 26,
 'b': 8,
 'c': 11,
 'd': 14,
 'e': 40,
 'f': 7,
 'g': 8,
 'h': 4,
 'i': 25,
 'k': 2,
 'l': 8,
 'm': 6,
 'n': 30,
 'o': 18,
 'p': 10,
 'r': 19,
 's': 15,
 't': 26,
 'u': 10,
 'v': 3,
 'w': 7,
 'x': 3,
 'y': 4}
```

In [7]:
```
# We can count the words too!
# for that split the string!
words = message.split()
print(words)
```

```
['JupyterLab', 'is', 'a', 'web-based', 'interactive', 'development', 'environ
ment', 'for', 'Jupyter', 'notebooks,', 'code,', 'and', 'data.', 'JupyterLab',
'is', 'flexible:', 'configure', 'and', 'arrange', 'the', 'user', 'interface',
'to', 'support', 'a', 'wide', 'range', 'of', 'workflows', 'in', 'data', 'scie
nce,', 'scientific', 'computing,', 'and', 'machine', 'learning.', 'JupyterLa
b', 'is', 'extensible', 'and', 'modular:', 'write', 'plugins', 'that', 'add',
'new', 'components', 'and', 'integrate', 'with', 'existing', 'ones.']
```

In [8]:

```
""" Split method split the string by the spaces.
You could have split it with any other arguments.
For example:---"""

example = "SplitABCbyABCotherABCarguments"
splitedList=example.split('ABC')
print(splitedList)
```

```
['Split', 'by', 'other', 'arguments']
```

In [9]:

```
# we already have a list called words - Line no: In[7]
countW = {} # Another way to initialize a dictionary
for word in words:
    countW.setdefault(word, 0)
    countW[word] += 1
pprint.pprint(countW)
```

```
{'Jupyter': 1,
 'JupyterLab': 3,
 'a': 2,
 'add': 1,
 'and': 5,
 'arrange': 1,
 'code,': 1,
 'components': 1,
 'computing,': 1,
 'configure': 1,
 'data': 1,
 'data.': 1,
 'development': 1,
 'environment': 1,
 'existing': 1,
 'extensible': 1,
 'flexible:': 1,
 'for': 1,
 'in': 1,
 'integrate': 1,
 'interactive': 1,
 'interface': 1,
 'is': 3,
 'learning.': 1,
 'machine': 1,
 'modular:': 1,
 'new': 1,
 'notebooks,': 1,
 'of': 1,
 'ones.': 1,
 'plugins': 1,
 'range': 1,
 'science,': 1,
 'scientific': 1,
 'support': 1,
 'that': 1,
 'the': 1,
 'to': 1,
 'user': 1,
 'web-based': 1,
 'wide': 1,
 'with': 1,
```

```
'workflows': 1,
'write': 1}
```

# Automate the typing!

In [10]:

```
# In the 5 sec window you have to place the cursor at any desired position
# Maybe in your friend's inbox :p
# For demonstration, I'll just put the cursor in the next row

import pyautogui as auto
import random
import time
message = ['hi','hello','hey', 'are you there!']
time.sleep(5)
for i in range(10):
    text = random.choice(message)
    auto.typewrite(text)
    auto.press('Enter')
```

In [ ]:

```
hi
hey
hi
hi
hi
hi
hello
are you there!
are youthere!
hey
```

# That's it for today! Try to solve the problem sheet!

Cheers, Happy coding :)