```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Lab1: Introduction to Jupyter Notebook and Python Review

In this lab you will learn how to use Jupyter notebook environment and review some basic concepts in python programming language. Each section contains exercises that you need to complete. We will start with simple concepts and data structure and work our way up to some simple tasks we need in handling data.

Jupyter Notebook

This page is a Jupyter notebook. You can write programs and run them to see the results, as well as writing text cells just like this one.

Text cells

Each rectangle containing text or code is called a *cell*. You can edit any text cells by double-clicking on them. They are written in Markdown format. If you are not familiar with the syntax, you can check the following link Markdown Syntax

After you edit a text cell, click the "run cell" button at the top that looks like ▶| or hold down `shift` + `return` to confirm any changes.

Code cells

Other cells contain code in the Python 3 language. Running a code cell will execute all of the code it contains.

To run the code in a code cell, first click on that cell to activate it. It'll be highlighted with a little green or blue rectangle. Next, either press ▶| or hold down `shift` + `return`.

Running this notebook

To run a Jupyter notebook, first navigate to the directory containing the downloaded notebook. From a command line, activate the environment and run:

(...) $ jupyter notebook

You need to to have Python 3 and Jupyter installed on your system. If you don't run the following command:

(...) $ conda install jupyter notebook

Submitting Lab assignments on Gradescope

Edit the following cell so that it prints "Hello World!" when run:

```
# Edit me to print "Hello World!"
```

Now that everything is running, let's start with some writing a simple function: **Exercise 1.1** Write a python function that returns the sum of squares in range [1, 2, ..., n]

```
def sqaure(n):
    totalSum = 0
    for i in range (0, n+1):
        totalSum = totalSum + (i*i)
    return totalSum

n = 5
print(sqaure(n))
```

```
55
```

Exercise 1.2 Write a lambda function that returns the sum of squares in range [1, 2, ..., n]

```
squares = lambda x: 1 if x == 1 else x*x + squares(x-1)
print(squares(5))
```

```
55
```

strings

Strings are a data type in Python for dealing with text, they are iterable, immutable.

Exercise 2.1 For a given string s, write a function that returns a new string that doubles each character of the original string. For instance, if the user enters *Pγthon*, the output should be *PPγγtthhoonn*.

```
def double_letters (s):
    n = s
    m = "_"
    for i in range(0, len(s)):
        m = m+s[i]+n[i]
    return m[1:]


s = 'Hello'
print(double_letters(s))
```

```
HHeellllloo
```

Python has a number of powerful features for manipulating strings.

Exercise 2.2 For the given string s, print the following:

1. The total number of characters in the string
2. The string repeated 5 times
3. The first character of the string
4. The first three characters of the string
5. The last three characters of the string

6. The string backwards
7. The string with its first and last characters removed
8. The string in all caps
9. The string with every a replaced with an e
10. Concatenate "The" to the string

```
s = 'aardvark'
```

```
s = 'aardvark'
print(len(s))
for i in range (0, 5):
    print(s)
print(s[0])
print(s[0:3])
print(s[len(s)-3:])
print(s[::-1])
print(s[1:len(s)-1])
print(s.upper())
print(s.replace("a", "e"))
print("The"+s)
```

```
8
aardvark
aardvark
aardvark
aardvark
aardvark
a
aar
ark
kravdraa
ardvar
AARDVARK
eerdverk
Theaardvark
```

Lists

The most fundamental data structure in Python is a list. A list is simply an ordered collection.

Exercise 3.1 For the list given below, do the following tasks:

1. print the first three item
2. print the last item
3. print the last three items
4. print the items with odd index
5. print the list in reversed, you should give at least two methods

```
days = ['sat','sun','mon','tue','wed','thu','fri']
```

```
print(days[:3])
print(days[len(days)-1:len(days)])
print(days[len(days)-3:len(days)])
for i in range (1, len(days)):
    if (i%2 != 0):
        print(days[i])
print(days[::-1])
days.reverse()
print(days)
```

```
['sat', 'sun', 'mon']
['fri']
['wed', 'thu', 'fri']
sun
tue
thu
['fri', 'thu', 'wed', 'tue', 'mon', 'sun', 'sat']
['fri', 'thu', 'wed', 'tue', 'mon', 'sun', 'sat']
```

List comprehensions are a powerful way to create lists:

```
l = [i for i in range(5)]
```

The example above will create a list L = [0,1,2,3,4] For the next exercise, use list comprehension:

Exercise 3.2

1. Generates a list L of 15 random numbers between 1 and 100
2. Replace each element in L with its square.
3. Count how many items in L are greater than 60.
4. Count how many items in L are less than 30.

```
import random

L = []

for i in range(0,15):
    L.append(random.randint(1,100))

print (L)

square = [n ** 2 for n in L]
L = square
print(L)

count = 0
for i in L:
    if i > 60 :
        count = count + 1
```

```
print(count)

count = 0
for i in L:
    if i < 30 :
        count = count + 1

print(count)
```

```
[47, 4, 15, 98, 11, 28, 52, 48, 71, 100, 75, 34, 82, 50, 46]
[2209, 16, 225, 9604, 121, 784, 2704, 2304, 5041, 10000, 5625, 1156,
6724, 2500, 2116]
14
1
```

Dictionaries

A fundamental data structure which associates values with keys and allows you to quickly retrieve the value corresponding to a given key.

Exercise 4.1 Raccoon University is holding an event to recognize distinguished alumni. For alumni receiving recognition, we have prepared a dictionary that contains their names and class of graduation:

```
rac_alum = {
    'Albert': '1996',
    'Ada' : '1998',
    'Barry': '1996',
    'Carlos': '1999',
    'Chris': '1996',
    'Claire': '1998',
    'Jill': '1999',
    'Rebecca': '2002',
    'Luis' : '2004',
    'Leon' : '1998',
}
```

As we are planning for the event, we recieve some corrections on our guests and we need to update the dictionary: A new guest has RSVP'd, an old guest has changed plans and will not attend, and a correction to one of our entries is necessary. For the given dictionary:

1. Add a new guest named Ashley who graduated in 2004
2. Remove Barry from the dictionary
3. Change Leon's class of graduation to 2004

```
rac_alum = {
    'Albert': '1996',
    'Ada' : '1998',
    'Barry': '1996',
    'Carlos': '1999',
```

```
    'Chris': '1996',
    'Claire': '1998',
    'Jill': '1999',
    'Rebecca': '2002',
    'Luis' : '2004',
    'Leon' : '1998',
}

rac_alum.update([("Ashley", "2004")])
print(rac_alum)

del rac_alum['Barry']
print(rac_alum)

rac_alum["Leon"] = 2004
print(rac_alum)
```

```
{'Albert': '1996', 'Ada': '1998', 'Barry': '1996', 'Carlos': '1999',
'Chris': '1996', 'Claire': '1998', 'Jill': '1999', 'Rebecca': '2002',
'Luis': '2004', 'Leon': '1998', 'Ashley': '2004'}
{'Albert': '1996', 'Ada': '1998', 'Carlos': '1999', 'Chris': '1996',
'Claire': '1998', 'Jill': '1999', 'Rebecca': '2002', 'Luis': '2004',
'Leon': '1998', 'Ashley': '2004'}
{'Albert': '1996', 'Ada': '1998', 'Carlos': '1999', 'Chris': '1996',
'Claire': '1998', 'Jill': '1999', 'Rebecca': '2002', 'Luis': '2004',
'Leon': 2004, 'Ashley': '2004'}
```

We want to set the tables based on class of graduation and assign guests to a table corresponding to their class of graduation. For this task, we need to reverse the rac_alum dictionary and create a new one that maps class of graduation to a list of alumni who graduated that year. **Exercise 4.2** Write a function that reverses the keys and values of a dictionary. *Example:*

```
reverse_dict({"Alice": "Chicago", "Bob": "Detroit", "Jenna":
"Chicago", "Adam":"Chicago"})
# => {"Chicago": ["Alice", "Jenna", "Adam"], "Detroit": ["Bob"]}

from collections import defaultdict
dct = {"Alice": "Chicago", "Bob": "Detroit", "Jenna": "Chicago",
"Adam":"Chicago"}
neel = defaultdict(list)
{neel[v].append(k) for k, v in dct.items()}
print(dict(neel))
```

```
{'Chicago': ['Alice', 'Jenna', 'Adam'], 'Detroit': ['Bob']}
```

Exercise 4.3 In the game Scrabble, each letter has a point value associated with it. We use the following dictionary for the letter values:

```
points = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
          'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
          'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
          'V':4, 'W':4, 'X':8, 'Y':4, 'Z':10}
```

Write a function that calculates the score of a given word. Use it to find the score of each word in the following list:

```
words =['airplane' , 'zebra' , 'statistics','algorithm']
```

```
def scrabble_score(word):
    total = 0
    for letter in word:
        total += points[letter]
    return total

points = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
          'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
          'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
          'V':4, 'W':4, 'X':8, 'Y':4, 'Z':10}

words =['airplane' , 'zebra' , 'statistics', 'algorithm']

for i in range(len(words)):
    words[i] = words[i].upper()

for i in range(len(words)):
    print(scrabble_score(words[i]))
```

```
10
16
12
15
```

We can use dictionaries to count how frequently certain words appear in a text. **Exercise 4.4** In this exercise we will find the frequency of words in *The yellow wallpaper* by *Charlotte Perkins Gilman*. This book and many others are available for free on Project Gutenberg. The following line of code will read the entire contents of a file containing *The yellow wallpaper* text and stores it in a string called *data*:

```
dat = open('Yellow_Wallpaper_PROJECT_GUTENBERG.txt').read()
```

In order to analyze this text do the following:

1. Turn the whole string to lowercase
2. Remove all the punctuations from the text (you can use "from string import punctuation")
3. Split the string into a list of individual words
4. Create a dictionary and count the frequency of words

5. Print the first 20 and the last 20 words in your dictionary. The words must be in alphabetical order

```
dat = open('Yellow_Wallpaper_PROJECT_GUTENBERG.txt').read()
```

```
---------------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_12128/118805602.py in <module>
----> 1 dat = open('Yellow_Wallpaper_PROJECT_GUTENBERG.txt').read()

~\anaconda3\lib\encodings\cp1252.py in decode(self, input, final)
     21 class IncrementalDecoder(codecs.IncrementalDecoder):
     22     def decode(self, input, final=False):
---> 23         return
codecs.charmap_decode(input,self.errors,decoding_table)[0]
     24
     25 class StreamWriter(Codec,codecs.StreamWriter):

UnicodeDecodeError: 'charmap' codec can't decode byte 0x9d in position
1436: character maps to <undefined>
```